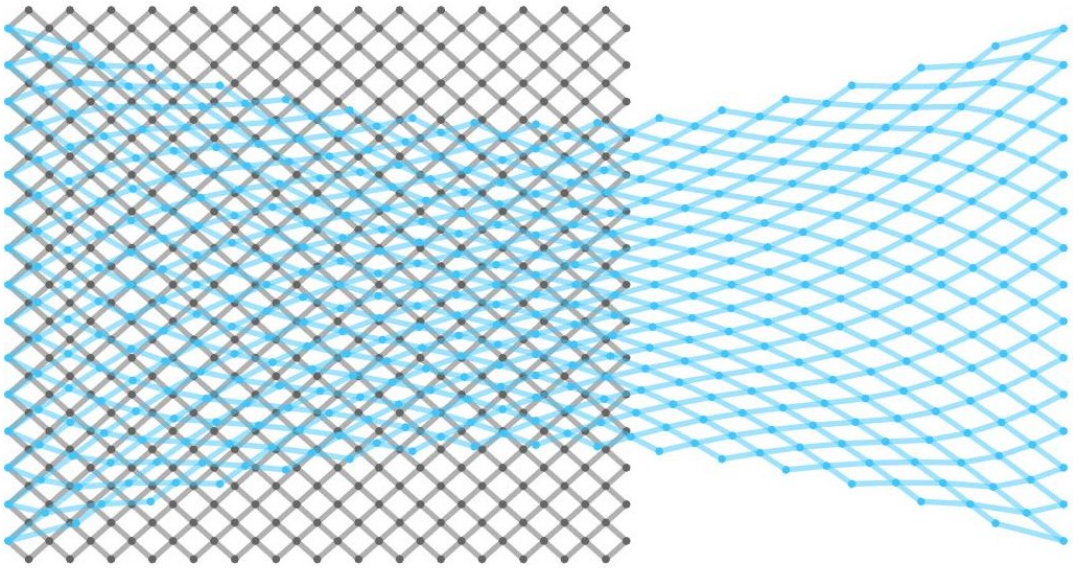

Numerical Simulation of Mechanical Properties of Fiber Network Models

Master's thesis in Applied Mathematics

ANDREA BERTL

DEPARTMENT OF MATHEMATICAL SCIENCES

UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2021
www.gu.se

Master's thesis 2021

Numerical Simulation of Mechanical Properties of Fiber Network Models

ANDREA BERTL

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Division of Applied Mathematics
University of Gothenburg
Gothenburg, Sweden 2021

Numerical Simulation of Mechanical Properties of Fiber Network Models
ANDREA BERTL

Supervisor I:	Axel Målqvist, Chalmers University of Technology, University of Gothenburg, Fraunhofer Chalmers Centre
Supervisor II:	Gustav Kettil, Fraunhofer Chalmers Centre
Examiner:	Anders Logg, Chalmers University of Technology, University of Gothenburg, Fraunhofer Chalmers Centre

Master's thesis 2021
Department of Mathematical Sciences
Division of Applied Mathematics
University of Gothenburg
SE-405 30 Gothenburg
Telephone +46 31 786 0000

Cover: Deformation of regular quadratic grid constructed in Matlab showing the original and displaced network.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Numerical Simulation of Mechanical Properties of Fiber Network Models
ANDREA BERTL
Department of Mathematical Sciences
University of Gothenburg

Abstract

In this thesis six models, q_1, q_2, t_1, t_2, h_1 and h_2, of discrete fiber networks are presented. They consist of regular rectangles, triangles and hexagons, whereby each shape is arranged in two different orientations.
Investigations on stiffness of networks for different choices of junction volume V_{ijl}, bending parameter κ_{ijl}, and material are carried out. Considering steel fibers, a constant glue parameter as junction volume, and grids of 20 to 40 meters of rod per square meter it is revealed that quadratic networks have the highest and lowest stiffness values, depending on their orientation. The stiffness of the hexagonal grids exceeds those of the triangular meshes.
By varying V_{ijl} and κ_{ijl} it is shown that hexagonal grids and the quadratic mesh without parallel lines to the axes, called q_2, are most affected by inner forces arising due to angular deviation. Therefore, it is concluded that angles change most within those networks.
Moreover, it is shown, that *Poisson's ratio* of the meshes increases with decreasing stiffness and vice versa.
Numerical evidence is provided to demonstrate that the grids, with exception of q_2, are less stiff subject to randomised node displacement. Without exception the stiffness values for all meshes decrease approximately linear if some arbitrary edges are removed.

Keywords: discrete fiber network model, linear elasticity, beam theory, stiffness, Poisson effect, Poisson's ratio.

Acknowledgements

I wish to express my gratitude to my supervisors Axel Målqvist and Gustav Kettil for their guidance during this project. Thank you both for your explanations, feedback, patience and endless ideas for alternatives whenever something did not work as intended.
I would also like to thank Malin and Rainer, not only for proofreading but also for your friendship, which I highly appreciate. Thanks to Elsa, Eric, Marvin and Jonas for the inspiring exchanges and pool nights.
Finally, my thanks go to my parents Rosa Maria and Helmut for your unconditional support and to you, Bernhard, for accompanying me in every situation, catching me and believing in me.

Andrea Bertl,
Gothenburg, August 2021

Contents

List of Figures

List of Tables

1 Introduction

This thesis aims to numerically model and compare mechanical properties of discrete fiber networks. Those consist of edges and nodes, representing fibers and bonds between them, respectively. Mass-spring models, where each edge has the properties of a spring, developing resistance forces when subjected to tensile or compression load, are a simple version [5]. Such models are an alternative to the finite element method to simulate elastic, continuous materials. They qualify to mimic actual lattices of connected fibers, such as truss systems in engineering. On a much smaller scale, lattice models can be used to represent matter on an atomic basis, where they originally come from [10]. Moreover, fiber network models are frequently used in graphic simulations and virtual reality applications to represent elastic objects [8]. Additionally, the models here are defined to develop spring-like forces counteracting angular changes. The theoretical foundations to create the discrete numerical model are introduced below. Thereafter the outline of the thesis is presented.

1.1 Basic Model

The general model consists of a two-dimensional discrete fiber network $(\mathcal{N}, \mathcal{E}, \mathcal{P})$, where $\mathcal{N}$ is the set of nodes, $\mathcal{E}$ the set of edges and $\mathcal{P}$ the set of edge pairs. In Figure 1.1 the network is illustrated as a grey rectangle. A tensile test with one clamped end of the object is modeled. Therefore, the nodes on the left hand side of the network are considered fixed, whereas the nodes of the right hand side, $i \in \mathcal{N}_{RHS}$, are displaced in x-direction and fixed in y-direction, those are marked dark and light blue in Figure 1.1, respectively. The displacement is defined by a vector $u \in \mathbb{R}^{2N}$ where N is the number of nodes. All nodes are designated with natural numbers $i \in \{1, \dots N\}$. For every node $i \in \mathcal{N}$, its displacement $\delta_i = (u_{i_x}, u_{i_y}) \in \mathbb{R}^2$ equals $u(2i-1, 2i)$ and describes the node's shift in x- and y-direction, respectively.

In the beginning, the x-displacement u_x for the nodes in $\mathcal{N}_{RHS}$ is set. Furthermore, the elasticity properties of the single fibers $(i, j) \in \mathcal{E}$, between nodes i and j, and pairs of fibers $(i, j, l) \in \mathcal{P}$, that are connected at one point j, are given. These features correspond to the material the fiber consists of. To ensure constant characteristics throughout the network, fibers are considered as cylindrical bars of a homogeneous, isotropic and linearly elastic material. This is used to construct the elasticity matrix $K \in \mathbb{R}^{2N \times 2N}$. Due to the equilibrium between external forces and internal resistance forces of the network it holds that:

$$F - Ku = 0.$$

Where $F \in \mathbb{R}^{2N}$ is the force vector necessary to obtain the given displacement. Utilising this relationship allows for computing F. Similar to the displacement vector, the entries $F(2i-1, 2i)$ correspond to the force in x- and y-direction applied at node i. F is depicted with red arrows in Figure 1.1. The displaced and deformed network is illustrated in Figure 1.2, with a maximal contraction of u_y parallel to the y-axis.

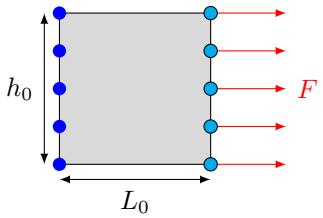

Figure 1.1: Basic setup of a discrete fiber network, with boundary conditions on the nodes at the left and right and side.

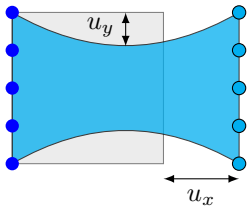

Figure 1.2: Deformed fiber network with respect to tensile load F acting on the right hand side and a fixed left hand side.

The sum of the forces applied on $i \in \mathcal{N}_{\text{RHS}}$ towards the x-direction is denoted f_x. This is used to compute the network's stiffness

$$s = \frac{\frac{f_x}{h_0}}{\frac{u_x}{L_0}},$$

together with the initial height $h_0 \in \mathbb{R}$ and length $L_0 \in \mathbb{R}$ of the network.

1.2 Outline

The following chapter deals with all necessary definitions and theorems from mechanics that are required to model the networks. Furthermore, the derivation of the elasticity matrix is explained. Thereafter follows a detailed description of the different network structures investigated and compared. Chapter 4 presents the numerical experiments of density-stiffness analyses, *Poisson's ratio* and random manipulations on grids, as well as their results. Finally, a conclusion is drawn and an outlook on possible future research is presented.

2
Background

The material and the structure of single fibers and pairs of connected fibers are crucial to determine the relationship of deformations and forces in a fiber network. Techniques to compute stress and strain of various members such as beams or springs are adduced in the field of mechanics of materials. In this chapter the basic mechanical definitions are presented. It is discussed, which cases are used in the experiments later on. The assembly of the elasticity matrix for a full network of structural members is described.

2.1 Mechanical Definitions

This section mainly traces back to Gere and Goodno [2].

2.1.1 Stress and Strain

The foundation of all carried out mechanical experiments is formed by external load applied to a structure. Such loads induce inner forces, i.e. *stress*, and measurable deformations or *strain*.

Definition 2.1.1. Stress, denoted by σ, is the force acting on a body to resist distorting external load. It equals the force F per unit area A.

$$\sigma = \frac{F}{A} \tag{2.1}$$

In the presented models it is assumed, that stress is uniformly distributed. The bars subject to stress are considered prismatic, which means that they have a unit cross section and a straight longitudinal axis in the neutral state, before any displacement. With these assumptions, stress can also be expressed as the force per cross sectional area A_c perpendicular to the external load:

$$\sigma = \frac{F}{A_c}. \tag{2.2}$$

Among the so-called *normal stresses*, i.e. stress acting perpendicular to the cut area, one differs between *tensile stress*, when the body is stretched and *compressing stress*, when it is squeezed. As explained in Chapter 1, the focus lies on the first and therefore also on *tensile strains* as a counterpart:

Definition 2.1.2. The **strain** ε is the degree of displacement evoked by external forces.

For normal stress the strain can alternatively be defined by the elongation δ per unit length L:

$$\varepsilon = \frac{\delta}{L}. \tag{2.3}$$

Definitions 2.1.1 and 2.1.2 hold for all kinds of materials as long as the following assumptions are fulfilled:

1. The deformation of the structure is uniform throughout its volume.
2. The structure is prismatic.
3. The loads act through the centroid of the cross sectional area.
4. The structure consists of homogeneous material, i.e. the mechanical properties are the same all over the structure.

If points 1. through 4. hold and stress as well as strain are normal, one speaks of *uniaxial stress* and *strain*.

The here presented models are not only restricted to uniaxial stress and strain but also to a special set of materials that are *linearly elastic*:

Definition 2.1.3. Elastic materials are not to permanently deformed until a certain external load has been reached. If furthermore the strain is linear proportional to the stress such materials are called **linearly elastic**.

2.1.2 Hooke's Law

The limitations described in the previous section allow to deduce a special, linear case of the law of elasticity. It describes the correlation of stress and strain in a body:

Hooke's Law

Theorem 2.1.4. *Axial stress and strain of linearly elastic materials are proportional to each other and are determined by the material's modulus of elasticity E in the following manner:*

$$\sigma = E\varepsilon. \tag{2.4}$$

The material-specific *modulus of elasticity* is also called *Young's modulus*. It is measured in Pascal and is high for stiff materials, e.g. steel. The modulus can slightly differ with regards to tension and compression.
Equation (2.4) only holds for uniaxial or longitudinal stress such as tension or compression of bars. Yet, this is enough for the numerical stretching models of fiber networks described in Section 2.2.2.1.

2.1.3 Poisson's Ratio

Naturally, if a bar is stretched or compressed by axial strain ε, the change of shape also implies lateral contraction or expansion ϵ.

Definition 2.1.5. The negative of the ratio of transverse strain to axial strain is called **Poisson's ratio** γ. For a bar subject to uniaxial stress it equals:

$$\gamma = -\frac{\epsilon}{\varepsilon}.$$

Poisson's ratio is the measure of the so called **Poisson effect**.

Within the linear elastic range of linearly elastic materials, *Poisson's ratio* is a constant. For most materials it lies within 0.25 and 0.35. If the loads get high enough for a material to enter the state of plastic deformation, γ can change drastically. Despite the proportionality of lateral and axial strain in a prismatic bar, further assumptions are needed to ensure that the lateral strain remains the same throughout the structural member under a fixed load:

1. The material has to be homogeneous.
2. The material has to be isotropic, i.e. it must have the same properties in all directions.

In Section 4.2 experiments are carried out to compare *Poisson's ratio* of several networks as a whole, not only for single fibers.

2.1.4 Density and Stiffness

After describing the assumptions for the materials, some definitions for the structure of the networks still remain open. As mentioned in Chapter 1, the edges of discrete networks are considered to be cylindrical bars, defined by their endpoints. The mechanical behaviour of fiber grids depends on the sizing, which in turn is derived from their *density*:

Definition 2.1.6. The **density** d of a fiber network is defined as the total edge length per unit area.

Let L_{ij} be the length of edge $(i,j) \in \mathcal{E}$, L the length of the network and h its height. Then the *density* is computed as:

$$d = \left(\sum_{(i,j)\in\mathcal{E}} L_{ij} \right) \frac{1}{Lh}. \tag{2.5}$$

For a unit edge length $a = L_{ij}$ for all edges $(i,j) \in \mathcal{E}$ and a fixed number of edges $m = |\mathcal{E}|$ the formula can be simplified to

$$d = \frac{am}{Lh}. \tag{2.6}$$

Furthermore, the grids are considered to be adjusted mass-spring models. Hereby, the fibers are interpreted as springs, which are mechanical objects storing energy.

The force used for displacement is a mass acting on the network of springs. Later on there will be set a fixed displacement for the right hand side of the grids. Therefore, the correlation of the actual elongation of springs and the required force to obtain the desired deformation matters:

Definition 2.1.7. The **stiffness** s of a structure is the resistance to deformations such as bending, twisting or stretching.

The choice of a linearly elastic material for the fibers entails linear proportionality between the deformation δ and the applied force F. This is determined by the stiffness value s:

$$F = s\delta. \tag{2.7}$$

Using Definitions 2.1.1 and 2.1.2 in Theorem 2.4 leads to the following formula:

$$\delta = \frac{FL}{EA_c}. \tag{2.8}$$

Here the denominator EA_c is called the *axial rigidity* of the body. From Equation (2.8) follows that the stiffness for a single spring is obtained by

$$s = \frac{F}{\delta} = \frac{EA_c}{L}. \tag{2.9}$$

When considering several spring-like fibers composed into a network $(\mathcal{N}, \mathcal{E}, \mathcal{P})$, the force F in Equation (2.9) is replaced by the total force acting on the displaced boundary f divided by the initial height h_0 of the structure. Instead of the simple displacement δ, the strain of the displacement is used, i.e. the total elongation ΔL over the initial length L_0:

$$s = \frac{\frac{f}{h_0}}{\frac{\Delta L}{L_0}} = \frac{fL_0}{h_0 \Delta L}. \tag{2.10}$$

In the following model, explained in Chapter 3, ΔL will be a fixed displacement u_x of all nodes on the right hand side of the network parallel to the x-axis. The force applied on the nodes $i \in \mathcal{N}_{RHS}$ will only be nonzero in x-direction, such that f is the sum of x-entries only, denoted by f_x. It can be derived through the formulas, that the *stiffness* for a spring of a homogeneous material and fixed dimensions is constant.

2.1.5 Beam Theory

Due to the connections between springs and the orientations of the fibers within the grid, the loads applied are not always parallel to the longitudinal axis of the bars. This will cause bending of the edges and vertical displacements of the nodes in the network model. If the structural member is subjected to inclined or declined load, this can be resolved into lateral and axial load. The latter was discussed previously.

To deal with the lateral load, i.e. load perpendicular to the axis of the fiber, *beams* are introduced.

Definition 2.1.8. Beams are structural components subject to lateral load.

A beam of length L with one fixed support and one free end is studied to explain the bending of beams as result of transverse load. Such beams are also called *cantilever beams*. It is assumed that the bar is initially symmetric about the xy-plane. If moreover all load vectors act on the same plane, the deflection of the beam is also limited to this plane, which is called *the plane of bending*. The deflection of a point x along the beam towards the lateral direction is denoted by $v(x)$.

Definition 2.1.9. The **deflection curve** is the curve the original longitudinal axis of the beam forms subject to lateral load.

The stresses and strains that result from beam deflection depend on the *curvature* of the *deflection curve*. To explain this concept a beam subject to *pure bending* is considered. In other words, it is assumed that no shear forces are acting on the beam. Therefore, the *bending moment* is constant.

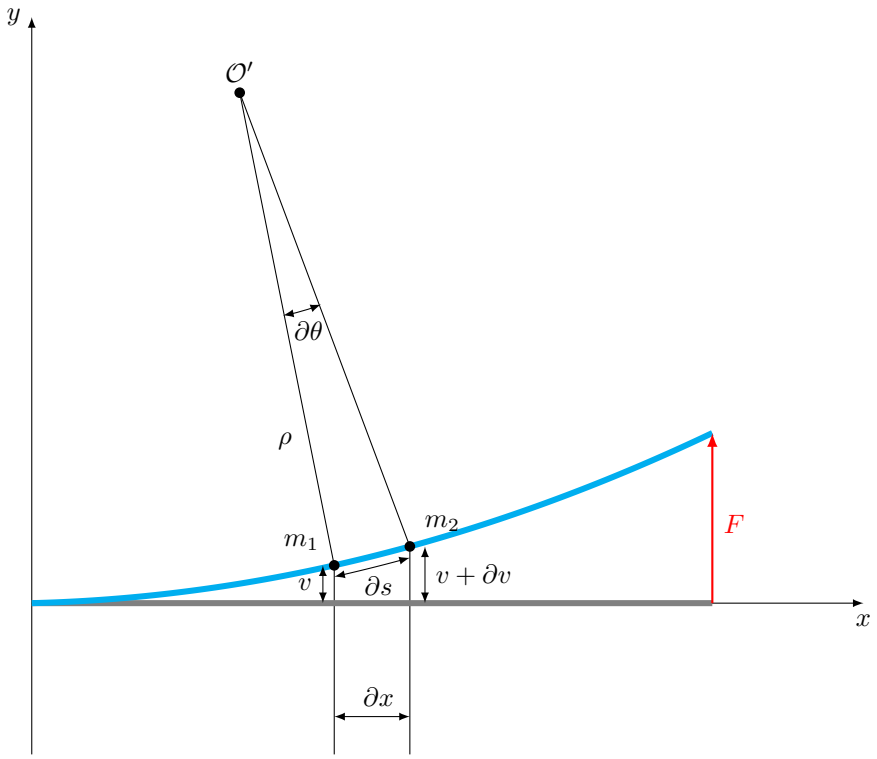

Figure 2.1: Geometrical deviation of the bending curvature κ of a cantilever beam, [2, p. 680].

As illustrated in Figure 2.1, two points m_1 and m_2 on the deflection curve with a small distance ∂s are chosen. The normals drawn to the tangent at m_1 and m_2 to

the deflection curve intersect at the *center of curvature* $\mathcal{O}'$. Whereas $\partial\theta$ describes the infinitesimal angle between the normals, the *radius of curvature* ρ is the distance from m_1 to the center of curvature.

Definition 2.1.10. The **curvature** κ of a deflection specifies how strong a beam is bent. It is the reciprocal of the *radius of curvature*:

$$\kappa = \frac{1}{\rho}. \tag{2.11}$$

Under some assumptions, a beam's deflection and the applied load are related according to the following theorem:

Euler-Bernoulli Beam Equation

Theorem 2.1.11. *Considering a prismatic, homogeneous, linearly elastic beam the distributed load q can be expressed as*

$$EI\frac{d^4v(x)}{dx^4} = -q \tag{2.12}$$

Where I is the moment of inertia and EI is the flexural rigidity.

The formula of the *moment of inertia* of a prismatic beam with cross sectional area A_c reads [2, p. 363]:

$$I = \int_{A_c} y^2 \partial A_c. \tag{2.13}$$

To proof the *Euler-Bernoulli Beam Equation* prior the *bending moment* and its behaviour towards *curvature* is introduced:

Definition 2.1.12. [13, p. 92, p. 167] The moment that results about the neutral axis at any section of a beam, caused by deflection forces, is called the **bending moment** M.

It is related to κ by the *Moment-Curvature-Equation*. A more detailed derivation of the formulas below can be found in [2, pp. 361]:

Moment-Curvature-Equation

Theorem 2.1.13. *According to the second equation of statics, the bending moment M equals the moment resultants of the normal stresses σ_x acting over the cross sectional area A_c,*

$$\partial M = -\sigma_x y \partial A_c.$$

Integrating over all elemental moments and formula (2.13) lead to the following expression of the curvature in terms of the bending moment:

$$\kappa = \frac{1}{\rho} = \frac{M}{EI}. \tag{2.14}$$

Now Theorem (2.1.11) can be proven, referring to Figure 2.1:

Proof (Euler-Bernoulli Beam Equation). Triangular geometry from the triangle $\mathcal{O}'$, m_1, m_2 yields the relation

$$\rho \partial \theta = \partial s.$$

If the bending effect is rather small and hence the deflection curve relatively flat, ∂s equals approximately the distance of its horizontal projection ∂x. Those observations lead to an alternative expression for κ:

$$\kappa = \frac{1}{\rho} = \frac{\partial \theta}{\partial s} \approx \frac{\partial \theta}{\partial x}.$$

Furthermore, assuming a small size of the angle θ between a horizontal line and the tangent at the deflection curve at m_1 it holds:

$$\theta \approx \tan \theta = \frac{\partial v(x)}{\partial x}.$$

Differentiation of θ towards x gives:

$$\kappa = \frac{\partial \theta}{\partial x} = \frac{\partial^2 v(x)}{\partial x^2}.$$

Making use of (2.14) it is also known:

$$\kappa = \frac{M}{EI} = \frac{\partial^2 v(x)}{\partial x^2}.$$

Next, an element of a beam under distributed load is investigated, as shown in Figure 2.2. Utilising the equilibrium of vertical forces acting on the sides of the beam element, the distributed load q can be expressed in terms of the shear forces, here denoted by τ:

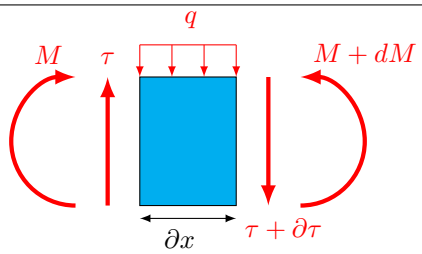

Figure 2.2: Element of a beam under distributed load q with shear forces τ and bending moments M acting on the sides, [2].

$$\frac{\partial \tau}{\partial x} = -q.$$

Similarly by the moment equilibrium it follows:

$$\frac{\partial M}{\partial x} = \tau.$$

These relations can be found in further detail in [2, pp. 320].

Under the assumption of a prismatic bar that has a unitary cross section, the flexural rigidity, EI is constant. With the equilibrium formulas above, it can finally be derived that:

$$\frac{\partial^2}{\partial x^2}\left(EI\frac{\partial^2 v}{\partial x^2}\right) = EI\frac{\partial^4 v}{\partial x^4} = -q.$$

□

2.1.6 Junction Volume

Bending fibers, theoretically explored in the previous section will lead to angular changes between two connected edges. Section 2.2.2.2 deals with the resulting stress and strain relationship in the network. The junction volume, V_{ijl}, defines the joint mass at the common node of two edges. It adds resistance to displacement due to angular deviation considering an edge pair $(i, j, l) \in \mathcal{P}$.

Throughout the experiments shown in Chapter 4 it is discovered that the computations for some networks are highly sensitive to the definition of the junction volume. Two ideas to define V_{ijl} are considered:

1. A constant glue parameter V_1 or
2. the comparison of the equation for angular deviation (Section 2.2.2.2) with the Euler-Bernoulli Beam Equation (2.12), V_2.

To ensure comparability the glue parameter for the first approach was adjusted to different patterns of grids. Looking at a central node j of the network, let p be the number of edge pairs in $\mathcal{P}$ with j as the connecting node. For regular grids, as presented in Chapter 3, p will be the same for all central nodes. Let w be the unit diameter of the fibers. Then the junction volume is chosen as:

$$V_1 = \frac{1}{p}w^3. \tag{2.15}$$

To derive V_2, Equation (2.12) is reconsidered for a positive load q such that a positive sign results on the right hand side, according to the sign convention of [2]. A cantilever beam is discretised with a path consisting of five nodes, which allows

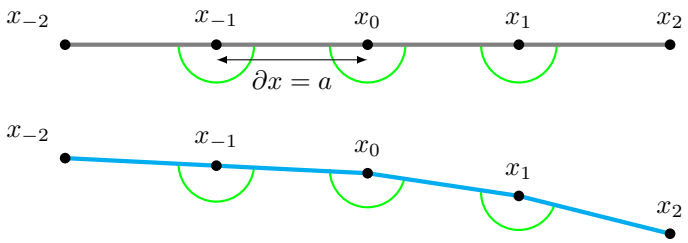

Figure 2.3: Displacement and edge pairs in a discretised cantilever beam, with its fixed end on the left hand side and subject to positive load on the right.

to investigate three edge pairs. According to the notation in Figure 2.3, those are $(x_{-2}, x_{-1}, x_0), (x_{-1}, x_0, x_1)$ and (x_0, x_1, x_2). Now the force $F^{II}_{x_0}$ arising in x_0 due to the angular change between the edges of the edge pairs is computed as stated later in Section 2.2.2.2:

$$\begin{aligned} F^{II}_{x_0} &= F^{II}_{x_0}(x_{-2}, x_{-1}, x_0) + F^{II}_{x_0}(x_{-1}, x_0, x_1) + F^{II}_{x_0}(x_0, x_1, x_2) \\ &= F^{II}_{x_0}(x_{-2}, x_{-1}, x_0) - F^{II}_{x_{-1}}(x_{-1}, x_0, x_1) - F^{II}_{x_1}(x_{-1}, x_0, x_1) + F^{II}_{x_0}(x_0, x_1, x_2) \\ &= \frac{\kappa V_{ijl}}{a^2}\left(v(x_{-2}) - 4v(x_{-1}) + 6v(x_0) - 4v(x_1) + v(x_2)\right). \end{aligned}$$

With the load being the distributed force $q = \frac{F}{\partial x}$ it holds that:

$$\begin{aligned} EI\frac{\partial^4 v(x_0)}{\partial x^4} &= q = \frac{F^{II}_{x_0}}{\partial x} \\ &= \frac{1}{\partial x}\frac{\kappa V_{ijl}}{a^2}\underbrace{\left(v(x_{-2}) - 4v(x_{-1}) + 6v(x_0) - 4v(x_1) + v(x_2)\right)}_{=:\varphi}. \end{aligned} \tag{2.16}$$

Furthermore, the central finite difference method for $v'''' = \frac{\partial^4 v(x_0)}{\partial x^4}$, reads:

$$v''''(x_0) = \frac{\partial^4 v(x_0)}{\partial x^4} = \frac{v(x_{-2}) - 4v(x_{-1}) + 6v(x_0) - 4v(x_1) + v(x_2)}{\partial x^4} + \mathcal{O}(\partial x^3).$$

The remainder term $\mathcal{O}(\partial x^3)$ is neglected, and it is noted, that the discretisation distance ∂x equals the edge length a in this model. Hence, by comparing expression φ with v'''' Equation (2.16) gets:

$$EI\frac{1}{a^4} = \frac{\kappa V_{ijl}}{a^3}$$

Equating the bending parameter with the moment of elasticity, $\kappa \equiv E$ gives the second approach used for the junction volume, $V_2 = V_{ijl}$:

$$V_2 = \frac{I}{a}. \tag{2.17}$$

2.2 Elasticity Matrix Assembly

In Equation (2.10) the force, length-change and dimensions of a network are necessary to determine the desired stiffness. The model presented here deals with fixed displacements and set dimensions. Still the force, necessary to achieve the desired displacement has to be determined. This procedure is described below.

2.2.1 Single Unconnected Rods

First m non-connected horizontal rods, of the same length L, which are fixed on one side and exploit to a tensile stress at the other side are studied. It is assumed that these rods are put next to each other such that the total height of the structure is h. According to Equation (2.5) the density of this arrangement is:

$$d_{rods} = \frac{mL}{hL} = \frac{m}{h}. \tag{2.18}$$

To displace the endpoint i of a single rod according to the displacement vector $u_i = (u_{i_x}, u_{i_y})$ the necessary force can be computed by Formula (2.7). Here a force parallel to the x-axis is chosen, such that $u_{i_y} = 0$ for all nodes $i \in \mathcal{N}_{RHS}$. Furthermore the x-displacement is set uniformly such that $\delta = u_{i_x} = u_x$. Moreover, applying the formula for the stiffness for a single fiber (2.9) yields:

$$F_{rod} = \frac{EA_c}{L} u_x. \tag{2.19}$$

Since (2.19) applies for all m rods, the total force is:

$$F_{rods} = mF_{rod} = m\frac{EA_c}{L} u_x.$$

The elasticity factor $\frac{EA_c}{L}$ is now called K for simplicity's sake. Moreover, the length change ΔL equals u_x and the stiffness for the whole structure of m rods can be computed according to (2.10):

$$s_{rods} = \frac{\frac{mKu_x}{h}}{\frac{u_x}{L}} = \frac{mKL}{h} = dA_cE. \tag{2.20}$$

2.2.2 Connected Networks

This part follows the results presented by Gustav Kettil et al. [5, Ch. 5], [4], [3]. The steps discussed in the previous section are now repeated for an actual network $(\mathcal{N}, \mathcal{E}, \mathcal{P})$. This is more complicated, since the displacement vector u only assigns fixed values to some nodes on the boundaries, whereas all other values have to be derived. Furthermore, from the connections between the single springs emerges not only tensional strain but also angular deviations. That gives rise to a second kind of inner force, counteracting angular change. A third inner force, originating from a *Poisson effect* is described at the end of this section, albeit it is neglected in the model later on. Moreover, *Young's modulus* is replaced by a more general stiffness parameter k_{ij} assigning stiffness properties to every edge $(i, j) \in \mathcal{E}$. The elasticity factor K in the case of a connected network is an elasticity matrix of dimension $2N \times 2N$ where N is again the number of nodes, $|\mathcal{N}|$. K is created by subtracting up to three different matrices K^I, K^{II} and K^{III}. Each of them, in turn, results from the assembly of local matrices, describing the occurring inner forces on edge pairs or single edges.

2.2.2.1 Edge Extension

The first of these matrices, K^I, is used to express the force that acts on edges and aims to restore the original edge length after deformation. As an external force is applied on the network and nodes are eventually moved, single fibers are extended or compressed. Due to the elasticity properties of the material an inner force emerges to restore the initial proportions as illustrated in Figure 2.4.

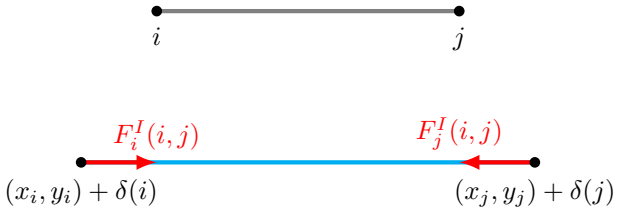

Figure 2.4: Illustration of edge extension and resulting force [5, p. 61].

For each edge (i, j) in the set of edges $\mathcal{E}$, connecting the nodes i and j of length L_{ij} a local matrix is computed. Here the stiffness parameter k_{ij} for a single edge is required. Let ΔL_{ij} describe the change of length of the edge under consideration, w_{ij} it's width, z_{ij} it's depth and d_{ij}^a the direction vector in the direction of $a \in \{i, j\}$,

$$d_{ij}^a = \frac{\begin{pmatrix} x_b \\ y_b \end{pmatrix} - \begin{pmatrix} x_a \\ y_a \end{pmatrix}}{\left| \begin{pmatrix} x_b \\ y_b \end{pmatrix} - \begin{pmatrix} x_a \\ y_a \end{pmatrix} \right|} \in \mathbb{R}^2.$$

Here x_a and y_a are the coordinates of node a and x_b, y_b of node $b \in \{i, j\}$, where $b \neq a$. The force vector $F_a^I(i, j)$ can be computed, according to the classical spring force model as it was seen before.

$$F_a^I(i,j) = k_{ij} \frac{w_{ij} z_{ij}}{L_{ij}} \Delta L_{ij} d_{ij}^a. \tag{2.21}$$

For studies of cylindrical fibers of diameter w_{ij} the formula for the cross sectional area $w_{ij} z_{ij}$ may be exchanged for the cylindrical area

$$A_c = \left(\frac{w_{ij}}{2}\right)^2 \pi.$$

The local matrix for edge (i, j), K_{ij}^I is then obtained by:

$$F_a^I(i,j) = K_{ij}^I \left(\{2a-1, 2a\}, \{1, \ldots, N\}\right) u,$$

for $a \in \{i, j\}$. Next, the assembly over all edges leads to K^I:

$$K^I = -\sum_{(i,j)\in\mathcal{E}} K_{ij}^I.$$

2.2.2.2 Angular Deviation

The second matrix, K^{II}, describes the force appearing to recover the original angle between two displaced edges. In Figure 2.5 the original edges are drawn grey, while the displaced edges that stem from moving nodes i by its displacement vector $\delta(i)$ and node l by $\delta(l)$ are marked blue. Once again K^{II} is obtained by assembly of local matrices. These are now defined on edge pairs (i, j, l) in the set of all edge pairs $\mathcal{P}$. Since parallel edges could be considered single long rods, they are neglected in the models throughout the thesis. $\mathcal{P}$ therefore consists of all triplets of nodes (i, j, l) such that $(i, j) \in \mathcal{E}$ and $(j, l) \in \mathcal{E}$, where not all x-positions nor all y-positions are equal. The central node describes the one node both edges of the edge pair have in common. Moreover, the network graph is considered undirected so that the edge pair (i, j, l) is equal to (l, j, i).

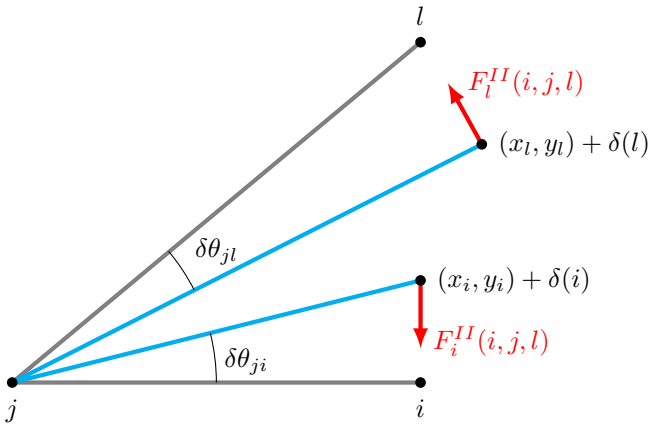

Figure 2.5: Illustration of angular deviation and resulting force [5, p. 62].

To compute the forces of angular deviation, the bending parameter κ_{ijl} and the junction volume V_{ijl} are necessary. These two parameters describe the resistance to bending an edge pair. Moreover, let $n_{ja}^j = d_{ja}^j \times \hat{z}$ denote the edge normal of the edge (j, a) where $a \in \{i, l\}$ and $\hat{z} = [0, 0, 1]^\top$. The angular change $\Delta\theta_{ijl}$ under the assumption of small angular changes is obtained by:

$$\begin{aligned}\Delta\theta_{ijl} &= \delta\theta_{ji} + \delta\theta_{jl} \\ &\approx \tan\delta\theta_{ji} + \tan\delta\theta_{jl} \\ &= \frac{(\delta_i - \delta_j)n_{ji}^j}{L_{ji}} + \frac{(\delta_l - \delta_j)n_{jl}^j}{L_{jl}}.\end{aligned}$$

As before, the force vector $F_a^{II}(i, j, l)$ for $a \in \{i, l\}$ can be computed:

$$F_a^{II}(i, j, l) = -\frac{\kappa_{ijl} V_{ijl} \Delta\theta_{ijl}}{L_{ja}} n_{ja}^j.$$

Those results yield the force vector for the central node $F_j^{II}(i, j, l)$:

$$F_j^{II}(i, j, l) = -F_i^{II}(i, j, l) - F_l^{II}(i, j, l).$$

Once again, this allows to derive the local matrix K^{II}_{ijl}:

$$F^{II}_a(i,j,l) = K^{II}_{ijl}\left(\{2a-1, 2a\}, \{1,\dots,N\}\right) u, \tag{2.22}$$

for $a \in \{i,j,l\}$. As a last step K^{II} is assembled:

$$K^{II} = -\sum_{(i,j,l)\in\mathcal{P}} K^{II}_{ijl}. \tag{2.23}$$

2.2.2.3 Poisson Effect

The third and last force considered corresponds to the *Poisson effect*. It describes the counteraction to compensate the total length change within an edge pair. As for the angular deviation, a local matrix K^{III}_{ijl} is computed for each edge pair $(i,j,l) \in \mathcal{P}$. If one of the edges is stretched or compressed, its width changes according to *Poisson's ratio* γ_{ijl} which is explained in Section 2.1.3. This causes inner forces in the second edge of the pair and vice versa. Figure 2.6 shows an edge pair (i,j,l) where only the blue edge is stretched but forces arise at both end nodes i and l.

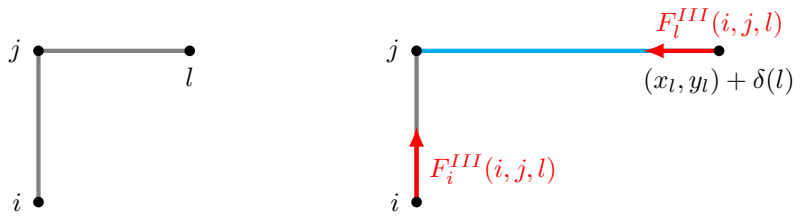

Figure 2.6: Illustration of the Poisson effect and its resulting force [5, p. 64].

Similarly to k_{ij} in the computation of K^I a stiffness parameter is required. Here however, it describes the stiffness for the whole edge pair and is denoted by η_{ijl}. This enables the calculation the force vectors corresponding to nodes i, j and l.

$$F^{III}_a(i,j,l) = -\eta_{ijl}\frac{w_{aj}z_{aj}}{L_{aj}}\left(\Delta L_{aj} + \gamma_{ijl}\frac{w_{bj}\Delta L_{bj}}{2L_{bj}}\left|n^j_{aj}\cdot d^j_{bj}\right|\right) d^j_{aj}$$
$$F^{III}_j(i,j,l) = -F^{III}_i(i,j,l) - F^{III}_l(i,j,l).$$

The last steps to obtain K^{III} are, mutatis mutandis, equivalent to (2.22) and (2.23).

2.2.3 Computation of the Stiffness of a Network

With the previous results the elasticity matrix K for the network can now be assembled:

$$K = K^I + K^{II} + K^{III}. \tag{2.24}$$

However, as it was mentioned before, in the following experiments K^{III} is neglected, since the correlation between the stiffness of single edges and edge pairs is not clear. Therefore, K in the following chapters stems from the formula

$$K = K^I + K^{II}. \tag{2.25}$$

In the next step the obtained elasticity matrix is adjusted to meet the boundary conditions, by replacing the columns corresponding to the nodes at the boundaries by the identity matrix. The new matrix is called K_{BC}. Moreover, a preliminary force vector $\hat{F}$ is created by setting all entries to 0 except the entries corresponding to the known displacement at the non-fixed side of the boundary. Now the displacement vector u for all nodes can be computed by solving the linear system of equations:

$$K_{BC} \cdot u = \hat{F}. \tag{2.26}$$

This allows to compute the actual force vector

$$F = Ku. \tag{2.27}$$

Now all entities needed to compute the stiffness according to (2.10) are known with $u_x = \Delta L$ being the maximal entry of u corresponding to x-displacement. It is to be noted, that the force, used in the formula (2.10) is not equal to F from (2.27) but obtained by summing up all relevant entries. When displacing the right hand side of the network towards the x-direction, i.e. all nodes $i \in \mathcal{N}_{RHS}$ that is:

$$f_x = \sum_{i \in \mathcal{N}_{RHS}} F(2i-1). \tag{2.28}$$

3

Fiber Networks

To model and compare grids with different patterns, a set of parameters has to be chosen. Starting with the quadratic structures, referred to as q_1 and q_2, the chapter then continues with explanations of the triangular and hexagonal meshes, called t_1, t_2, h_1 and h_2, respectively. Secondly, the definitions of base units for some of the grids are described, aiming to verify first results of the stiffness computations. Thereafter, the selection of units required for the analysis of stiffness values is explained.

3.1 Network Patterns

For each of the patterns two orientations of the basic form were chosen to simulate deformations caused by uniaxial forces in different directions. Dependant on the orientation care is taken to ensure that the networks are symmetrical regarding the x-axis. This is to avoid irregular displacements in y-direction when applying force in x-direction. However, the second triangular grid t_2 is non-symmetric for an odd number of rows, whereas the second hexagonal grid h_2 is non-symmetric for an even number of rows. The choice of 30 rows is motivated in Section 3.3 and solves the problem for t_2. In the same section it is clarified how the symmetry is obtained for h_2. All patterns are chosen to be regular, with a unit edge length a.

Boundary conditions are set at the left and right hand side. The left nodes, marked dark blue in Figures 3.1 through 3.6 are fixed, i.e. these nodes are not displaced. The rightmost nodes, marked light blue, are exposed to a horizontal force vector F, parallel to the x-axis and strong enough to displace those nodes according to a displacement vector u.

Edge pairs, necessary to compute the resulting elasticity behaviour from angular changes, were defined between any two adjacent, non-parallel edges. In the following Figures 3.1 through 3.6 edge pairs are marked with green arcs between the two paired fibers. Those arcs are of $90°, 60°$ and $120°$ for the quadratic, triangular and hexagonal patterns, respectively.

3.1.1 Quadratic Grids

Figures 3.1 and 3.2 display the two implemented quadratic setups. The first consists of rods, parallel to the x- and y-axis, the second of squares rotated by $45°$.

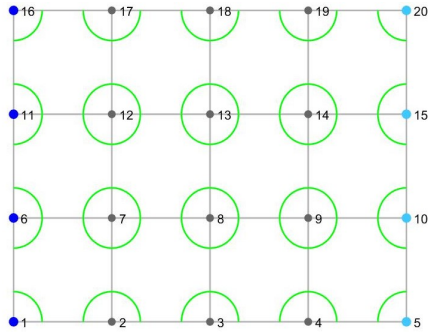

Figure 3.1: The network q_1 with 4 columns and 3 rows.

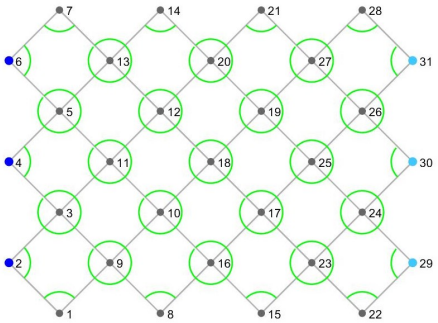

Figure 3.2: The network q_2 with 4 columns and 3 rows.

Knowing the number of columns c and rows r, the number of edges m can be derived. Furthermore, if the density d is given, the edge length a results from Formula (2.5). Finally, utilising the geometries of the grids, their respective heights h_0 and lengths L_0 are obtained. Those results allow the computation of the stiffness applied in Chapter 4.

Formulas q_1

$$
\begin{aligned}
m &= 2cr + c + r \\
a &= \frac{m}{dcr} \\
h_0 &= ar \\
L_0 &= ac.
\end{aligned}
$$

Formulas q_2

$$
\begin{aligned}
m &= 4cr \\
a &= \frac{m}{2d\left(c - \frac{1}{2}\right)\left(r - \frac{1}{2}\right)} \\
h_0 &= \sqrt{2}a\left(r - \frac{1}{2}\right) \\
L_0 &= \sqrt{2}a\left(c - \frac{1}{2}\right).
\end{aligned}
$$

3.1.2 Triangular Grids

The first triangular fiber composition consists of equilateral triangles with one edge parallel to the y-axis, the second of triangles rotated by 90°, with one edge parallel to the x-axis. Examples for both cases are shown in Figures 3.3 and 3.4.

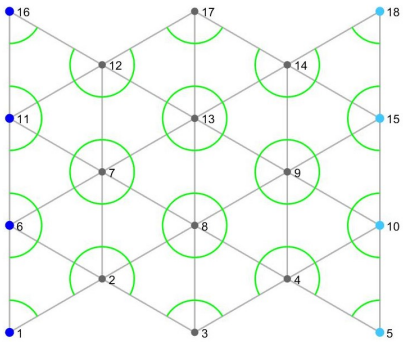

Figure 3.3: The network t_1 with 4 columns and 3 rows.

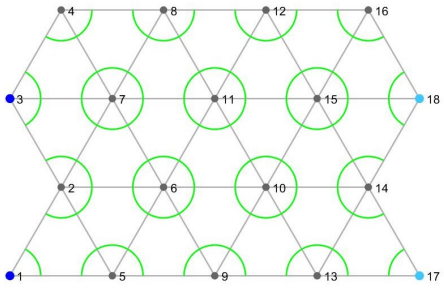

Figure 3.4: The network t_2 with 4 columns and 3 rows.

Similarly to the quadratic grids the network specific parameters are derived from its geometries. The number of edges for both triangular setups depends on an even or odd number of columns or rows. This is resolved by using ceiling and floor functions in the formulas for m.

Formulas t_1

$$
\begin{aligned}
m &= 3cr + r - c - 1 + \left\lceil \frac{c+1}{2} \right\rceil \\
a &= \frac{2m}{d\sqrt{3}\left(cr - \frac{c}{2}\right)} \\
h_0 &= ar - \frac{a}{2} \\
L_0 &= \frac{\sqrt{3}}{2} ac.
\end{aligned}
$$

Formulas t_2

$$
\begin{aligned}
m &= 2cr + c\left\lceil \frac{r+1}{2} \right\rceil \\
&\quad + (c-1)\left\lfloor \frac{r+1}{2} \right\rfloor \\
a &= \frac{2m}{d\sqrt{3}\left(cr - \frac{r}{2}\right)} \\
h_0 &= \frac{\sqrt{3}}{2} ar \\
L_0 &= ac - \frac{a}{2}.
\end{aligned}
$$

3.1.3 Hexagonal Grids

Last but not least, examples for both regular hexagonal networks are shown in Figures 3.5 and 3.6.

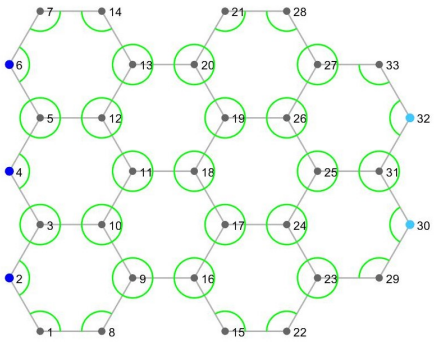

Figure 3.5: The network h_1 with 4 columns and 3 rows.

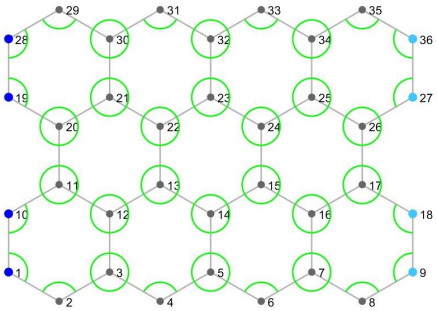

Figure 3.6: The network h_2 with 4 columns and 3 rows.

Once again the number of edges, the unit edge length, the height and the length of the networks can be obtained using the number of columns and rows as well as the density and the given geometries. As for the triangular meshes, the amount of edges changes with an odd or even number of columns or rows. Therefore ceiling and floor functions are included. To simplify the notation auxiliary variables $\hat{m}_1$ and $\hat{m}_2$ are introduced first.

Formulas h_1

$$\hat{m}_1 = (5r+1)\left\lceil \frac{c}{2} \right\rceil + r\left\lfloor \frac{c}{2} \right\rfloor$$

$$m = \begin{cases} \hat{m}_1, & \text{if } c \text{ odd} \\ \hat{m}_1 + 2(r-1), & \text{otherwise} \end{cases}$$

$$a = \frac{4m}{d\sqrt{3}\,(6cr - 3c)}$$

$$h_0 = a\sqrt{3}\left(r - \frac{1}{2}\right)$$

$$L_0 = \frac{3}{2}ac.$$

Formulas h_2

$$\hat{m}_2 = (5c+1)\left\lceil \frac{r}{2} \right\rceil + c\left\lfloor \frac{r}{2} \right\rfloor$$

$$m = \begin{cases} \hat{m}_2, & \text{if } r \text{ odd} \\ \hat{m}_2 + 2(c-1), & \text{otherwise} \end{cases}$$

$$a = \frac{4m}{d\sqrt{3}\,(6cr - 3r)}$$

$$h_0 = \frac{3}{2}ar$$

$$L_0 = a\sqrt{3}\left(c - \frac{1}{2}\right).$$

3.2 Base Unit Calculations

For sake of verifying the implemented grids and formulas, some patterns, namely q_1, t_1 and t_2 are investigated more closely. A repetitive part of the basic form, which is referred to as a *base unit*, is extracted. Then the stiffness of a grid of base units is computed, taking only the edge extension into account, i.e. setting the elasticity matrix to be $K = -K^I$, such that the resulting stiffness values would be comparable to single stretched rods, described in Section 2.2.1.

3.2.1 One Quadratic Grid

The base unit for q_1 consists of two edges and three nodes. If it is mirrored on the axis $y = x$, the original shape of a square results. The left nodes of the base unit are fixed, while the right node, marked as node 3 in Figure 3.7 is subjected to a force parallel to the x-axis. The edge $(2, 3)$, marked blue, therefore gets stretched and node 3 is displaced by $(x, 0)$.

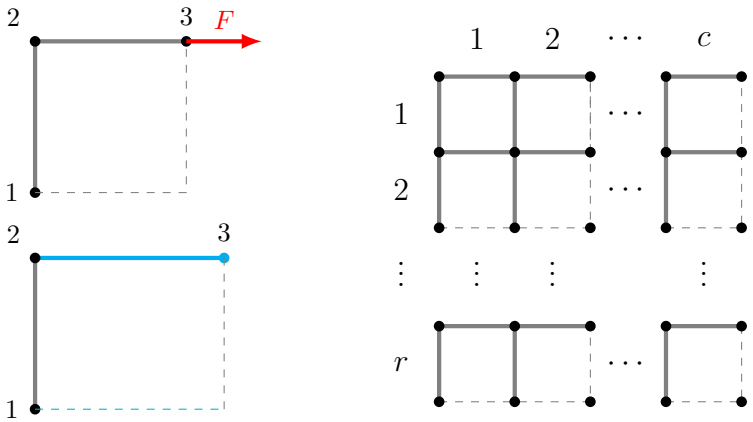

Figure 3.7: Quadratic base unit of pattern q_1.

The displacement $u_x(i)$ of node $i \in \{1,2,3\}$ is 0 towards the x-direction, for nodes 1 and 2 and non-zero for node 3, i.e. $u_x(3) = x$. Together with $u_y(i) = 0$ for all $i \in \{1,2,3\}$ the displacement vector u can be assembled for a single base unit as:

$$\left.\begin{aligned} u_x &= [0,0,x]^\top \\ u_y &= [0,0,0]^\top \end{aligned}\right\} \Rightarrow u \coloneqq [0,0,0,0,x,0]^\top.$$

The only stretched edge is $(2,3)$ and the resulting force F_{BU} corresponding to one base unit and considering the edge extension can be computed using Formula (2.21):

$$F_{\mathrm{BU}} = F_{32}^I = k\frac{A_c}{a}\Delta L_{23} d_{23}^3 = k\frac{A_c}{a}\begin{bmatrix} x \\ 0 \end{bmatrix}.$$

Considering the network on the right hand side of Figure 3.7 and imagining its completion, it is clear that the force of the whole network is obtained by multiplying F_{32}^I with $(r+1)$, where r is the number of rows. It suffices to consider the x-entry of F_{32}^I, denoted by F_x, since the y-displacement equals 0:

$$F = (r+1)F_x = (r+1)k\frac{A_c}{a}x.$$

The number of edges m of the network is obtained by taking cr times the number of edges of the base unit, namely two. To get the complete quadratic grid, as shown in Figure 3.1, $c+r$ edges have to be added. Inserting m, L_0 and h_0 as denoted in the box of formulas for q_1 into Equation (2.6) for the density yields:

$$d = \frac{(2cr+c+r)a}{cra^2} = \frac{2cr+c+r}{cra} \tag{3.1}$$

For an infinite amount of columns and rows, Equation (3.1) gets the same as it was for the incomplete network with $2cr$ edges:

$$\lim_{c,r\to\infty} d = \frac{2}{a}. \tag{3.2}$$

Substituting the previous results into Equation (2.10) and letting the number of rows rise to infinity again, the stiffness for q_1 is obtained. In the final step (3.2) is used:

$$\begin{aligned} s &= \frac{F_x(r+1)ca}{crax} = \frac{k\frac{A_c}{a}x(r+1)ca}{crax} = kA_c\frac{r+1}{ra} \\ \lim_{r\to\infty} s &= kA_c\frac{1}{a} = \frac{kA_cd}{2}. \end{aligned} \tag{3.3}$$

3.2.2 The Triangular Grids

The triangular nets are treated quite similar to the square pattern. The only difference is that the mesh of base units is not completed and the convergence behaviour is not used. Instead, it is shown in Section 3.2.3 that the stiffness values of the triangular grids from Section 3.1.2 approach those of the networks defined below. The latter consist of of base units.

3.2.2.1 Triangular Shape t_1

For the first triangular orientation, with one edge parallel to the y-axis, the base unit consists of two edges and three nodes. Using the x-axis as the mirror axis, the base form is restored. As for the quadratic grid, only node 3, marked in Figure 3.8, is displaced and only the blue edge, $(2, 3)$, changes in length by applying force F.

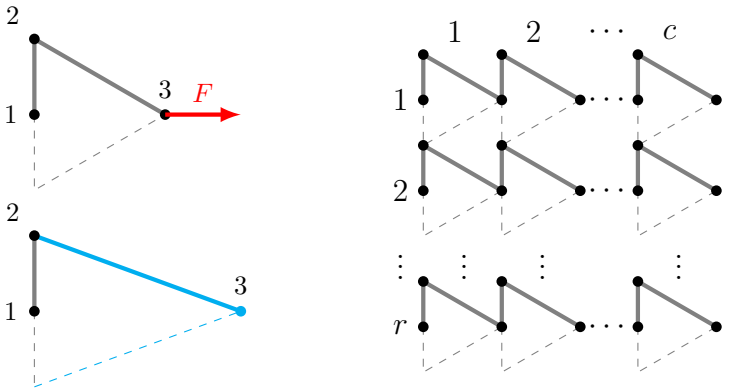

Figure 3.8: Triangular base unit of pattern t_1.

The displacement vector for the base unit of t_1 is equal to the one for the quadratic base unit:

$$\left.\begin{aligned} u_x &= [0,0,x]^\top \\ u_y &= [0,0,0]^\top \end{aligned}\right\} \Rightarrow u = [0,0,0,0,x,0]^\top.$$

The force vector of the base unit, F_{BU}, results again from the calculation of F^I for the single stretched edge, taking the geometry of the base unit into account:

$$F_{\mathrm{BU}} = F^I_{32} = k\frac{A_c}{a}\Delta L_{23} d^3_{23} = k\frac{A_c}{a} x \begin{bmatrix} \frac{3}{4} \\ 0 \end{bmatrix}.$$

By neglecting the zero valued y-entries and multiplying F_{BU} with the number of rows, the force for the network consisting of base units is obtained:

$$F = rF_x = rk\frac{A_c}{a}x\frac{3}{4}.$$

The total edge length per base unit is $(a + \frac{a}{2})$. This occurs cr times in the network. Length and height of the base unit are $\frac{\sqrt{3}a}{2}$ and $\frac{a}{2}$. For the network's dimensions one has to multiply this by the number of columns and rows, respectively. Inserting those results into the formula of the density gives:

$$d = \frac{cr\left(\frac{a}{2} + a\right)}{Lh} = \frac{\frac{3a}{2}}{cra^2\frac{\sqrt{3}}{4}} = \frac{2\sqrt{3}}{a}.$$

Formula (2.10) is used again with all results from above to calculate the stiffness for the network displayed in Figure 3.8 as a function of the elasticity parameter k, the cross sectional area A_c and the density d:

$$s = \frac{rF_x(c+1)c\frac{\sqrt{3}}{2}a}{cr\frac{a}{2}x} = rk\frac{A_c}{a}x\frac{3}{4}\frac{\sqrt{3}}{x} = kA_cd\frac{3}{8}. \tag{3.4}$$

3.2.2.2 Triangular Shape t_2

The last considered base unit corresponds to the rotated triangular pattern. Both, the base unit and the network it entails are illustrated in Figure 3.9. Again it consists of two edges and three nodes. The respective mirror axis is the y-axis.

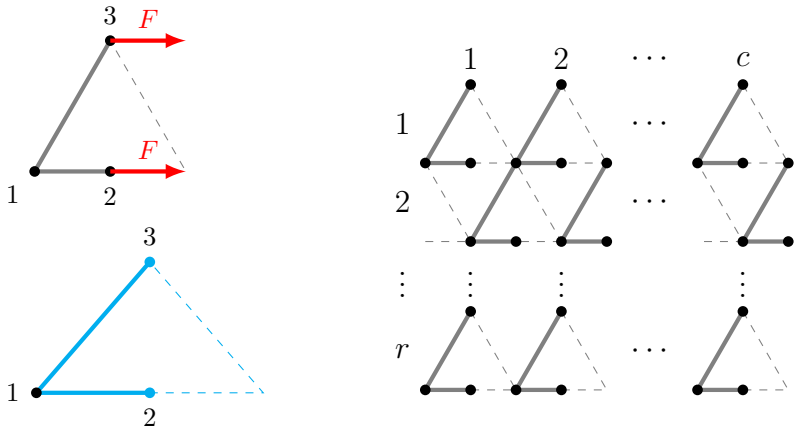

Figure 3.9: Triangular base unit of pattern t_2.

In this base unit two nodes are located on the right hand side and have a nonzero x-displacement:

$$\left.\begin{array}{l} u_x = [0, x, x]^\top \\ u_y = [0, 0, 0]^\top \end{array}\right\} \Rightarrow u = [0, 0, x, 0, x, 0]^\top.$$

Therefore, both edges of the unit get stretched and F_{ij}^I is computed for them by:

$$F_{12}^I = k\frac{A_c}{a}\Delta L_{12}d_{12}^2 = k\frac{A_c}{a}2x\begin{bmatrix}1\\0\end{bmatrix},$$
$$F_{13}^I = k\frac{A_c}{a}\Delta L_{13}d_{13}^3 = k\frac{A_c}{a}x\begin{bmatrix}\frac{1}{4}\\\frac{3}{4}\end{bmatrix}.$$

Once again the zero entries corresponding to the y-displacement are neglected. The force vector for the base unit is obtained by adding up the local force vectors for both edges:

$$F_x = F_{12}^I + F_{13}^I = k\frac{A_c}{a}x\frac{9}{4}.$$

Multiplying this result by r returns the force vector for the network of r rows of base units:

$$F = rF_x = rk\frac{A_c}{a}x\frac{9}{4}.$$

The length of the base unit corresponds to the height of the base unit of t_1 and vice versa. Again for the network those values have to be taken times c and r. The total edge length is computed exactly as for t_1:

$$d = \frac{cr\left(\frac{a}{2}+a\right)}{Lh} = \frac{2\sqrt{3}}{a}.$$

In the final step, the stiffness is computed by using the results from above and Formula (2.10) once more:

$$s = \frac{rF_x c\frac{a}{2}}{cr\frac{\sqrt{3}}{2}ax} = \frac{F_x}{\sqrt{3}x} = \frac{k\frac{A_c}{a}x\frac{9}{4}}{\sqrt{3}x} = \frac{kA_c 9}{a4\sqrt{3}} = kA_c\frac{3}{8}\frac{2\sqrt{3}}{a} = kA_c q\frac{3}{8}. \tag{3.5}$$

3.2.3 Results

Figure 3.10 depicts the results of the base unit calculations in comparison to the stiffness values for the same mechanical parameters. The used elasticity matrix is $K = K^I$.

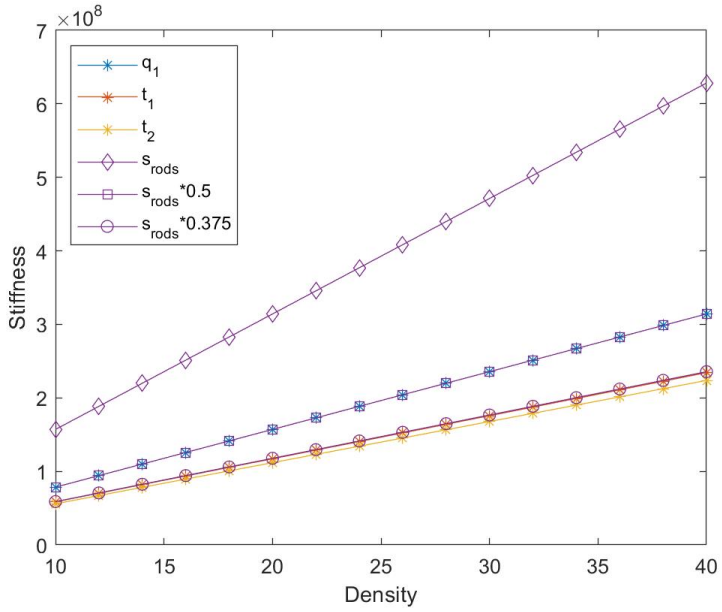

Figure 3.10: Comparison of stiffness calculations for the base units of q_1, t_1 and t_2 with s_{rods}. The stiffness of single rods according to Formula (2.10) is drawn with purple diamonds. The stiffness computed for quadratic grid equals the value of $s_{rods} \cdot 0.5$ according to (3.3) and marked by purple circles. Also the triangular networks approach stiffness values very close to $s_{rods} \cdot 0.375 = \frac{3}{8}s_{rods}$ from equations (3.4) and (3.5), depicted as purple circles.

Even though it does not make sense in a realistic setup, for this experiment the y-displacements for all nodes are set to 0 to avoid singular matrices.

If the networks consisting of the triangular base units are completed as indicated by the dashed lines in Figures 3.8 and 3.9, resulting grids include edges of length $\frac{a}{2}$ and incomplete triangles. To avoid this, the triangular networks were changed to the networks presented in Figures 3.3 and 3.4. Hereby arises the difference between the lines for t_1, t_2 and the stiffness of the triangular base units, $s_{rods} \cdot 0.375$, visible in Figure 3.10.

3.3 Choice of Parameters

To compute the elasticity matrix $K = K^I + K^{II}$, material-specific parameters k_{ijl} and κ_{ijl} and reasonable dimensions for the fibers must be chosen. The edge length depends on the density, according to the formulas listed for all patterns in Section 3.1. Therefore, a scope D of density values has to be determined such that it results in a realistic value of L_{ij} for the simulated materials, regardless which pattern is considered. Furthermore, the number of rows and columns must be known to generate the grids. All parameters are considered constant throughout the whole networks, such that $k_{ij} = k, \kappa_{ijl} = \kappa, L_{ij} = a$ and $w_{ij} = w$ for all edges $(i, j) \in \mathcal{E}$ and all edge pairs $(i, j, l) \in \mathcal{P}$.

The stiffness parameter k basically describes the same as *Young's modulus* and is therefore equated with E. Also the bending parameter κ is defined in terms of E. If nothing else is stated explicitly, the relation $\kappa = 1 \cdot E = k$ is assumed. Only in Section 4.1.1 κ got scaled down. Hence, to determine k and κ it is enough to know the *modulus of elasticity*.

This thesis is limited to a choice of three materials, steel, carbon and paper. *Young's modulus* E, *Poisson's ratio* γ, a realistic diameter w and edge length a of fibers are listed in Table 3.1. For some parameters only the value chosen for the experiments in Chapter 4 is listed in the table, albeit the values can be within the range stated in the references.

Table 3.1: Material specific parameters of steel, carbon and paper, according to [2, p. 961], [6, p. 6], [7, p. 1], [9, pp. 39] and [12, p. 117].

Material	E in GPa	γ	w in m	a in m
Steel	200	$0.27 - 0.30$	0.01	0.1
Carbon	240	$0.26 - 0.28$	0.007×10^{-3}	0.04
Paper	10.5	$0.10 - 0.57$	0.018×10^{-3}	0.002

The number of rows and columns need to be large enough to ensure a stable stiffness value for any given density. At the same time the computational effort should be as small as possible to avoid too long run-times. Therefore, the stiffness of all six networks is plotted for a fixed density value and all materials. In Figure 3.11 the result for one such investigation is shown. The shapes of the graphs were quite similar for other choices of density and other materials. Since all curves are rather flat for values above $c = r = 30$, this exact amount of rows and columns was chosen for the experiments following in Chapter 4.

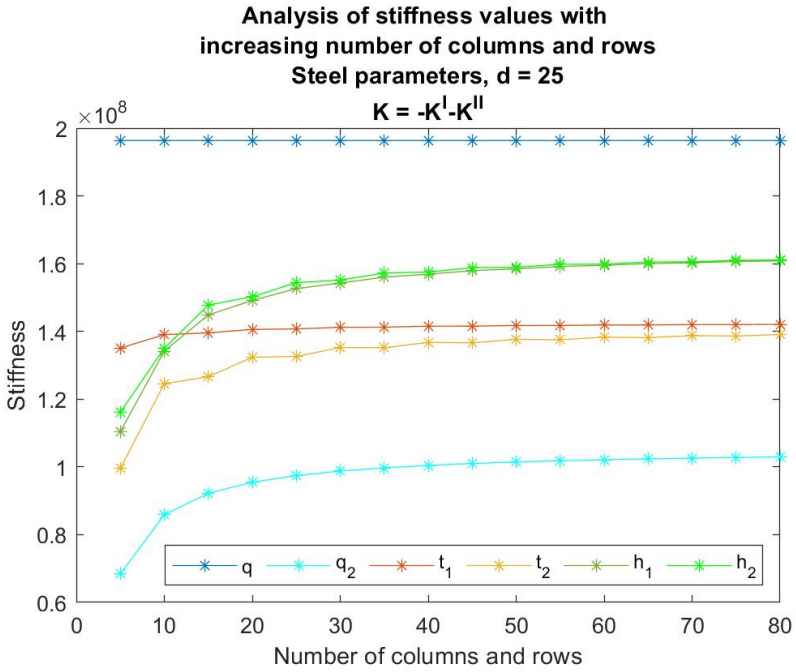

Figure 3.11: Stiffness of all grids in terms of the number of rows r and columns c.

However, as mentioned before, the second hexagonal grid, h_2, is not symmetric about the x-axis for an even amount of rows. As an example a fixed number of 30 columns and rows, steel parameters and a density of $d = 12.0602$ is considered, which results in the realistic edge length of $a = 0.1$. Hereby the maximal positive y-displacement of a node is 0.0186, whereas the maximal negative y-displacement is -0.0209. Even though the difference in the computed stiffness values were not high in comparison to the other network patterns, the maximal y-displacement matters for *Poisson's ratio* as seen in Section 4.2. Therefore, the number of rows and columns for h_2 was changed to 31, such that the y-displacement is symmetric about the x-axis. Compared to the same network with $c = r = 30$ this change evokes a different amount of edges m and hence slightly different values of the height, length and a marginally distinct edge length a. Since both, the stiffness values and *Poisson's ratio* are formulated as functions of the density, comparability to the other patterns remains.

For 30 or, in the case of h_2, 31 rows and columns, Table 3.2 lists the density values required to achieve the respective edge lengths from Table 3.1.

Table 3.2: Densities corresponding to realistic edge lengths for all patterns.

Pattern	Steel	Carbon	Paper
q_1	20.6667	51.6667	1.0333×10^3
q_2	20.6837	51.7093	1.0342×10^3
t_1	35.4239	88.5597	1.7712×10^3
t_2	35.4239	88.5597	1.7712×10^3
h_1	12.0602	30.1505	603.0103
h_2	12.0538	30.1346	602.6914

This results in the choice of $D = [10 : 40], D = [25 : 90]$ and $D = [590 : 1800]$ for steel, carbon and paper fibers, respectively.

4
Results

The formulas and the settings from the previous chapters allow the comparison of mechanical properties. Since most effects described are best visible using the parameters of steel fibers, the focus lies on this material. Even though the numerical values differ, most effects described are similar for other choices of materials. This chapter starts off by analysing the development of stiffness values of the previously presented choices of networks. Thereafter, those results are compared to the second possible choice of the junction volume, mentioned in Section 2.1.6. In a third section the other two materials, carbon and paper are studied. Then *Poisson's ratio* is presented for the different patterns and finally the effect of random changes to the grids are shown.

4.1 Density-Stiffness Analysis

One of this project's objectives is to determine which patterns result in the most and least stiff networks. Generally, the stiffness of networks increases at a rising density, as more edges and edge pairs contribute to higher resistance to external force. Nevertheless, another question is how the relationship between the stiffness of the meshes would change with increasing density. To answer both, the stiffness is expressed as a function of density and calculated for all six patterns presented in Section 3.1. Since the definitions of the different patterns lead to distinct dimensions at the same density, the displacement u_x towards the x-direction at the right hand side is adjusted to the original length L_0 of the grid,

$$u_x = \frac{\alpha}{100} L_0,$$

where α is an arbitrary positive number.

Realistic displacements are relatively small and deformations become less visible. Therefore, a rather large displacement of $\alpha = 50$ is chosen for the grids in Figure 4.1 to better illustrate how the networks evolve subject to a load pulling parallel to the x-axis.

The actual experiment to compare the stiffness of fiber networks is performed for a value of $u_x = 0.1L_0$. All parameters, including the settings for the density, are chosen for steel as presented in Tables 3.1 and 3.2. A constant glue parameter was chosen for the junction volume V_1, according to Equation (2.15), with $p = 4$ for the quadratic, $p = 6$ for the triangular and $p = 3$ for the hexagonal setups. This results

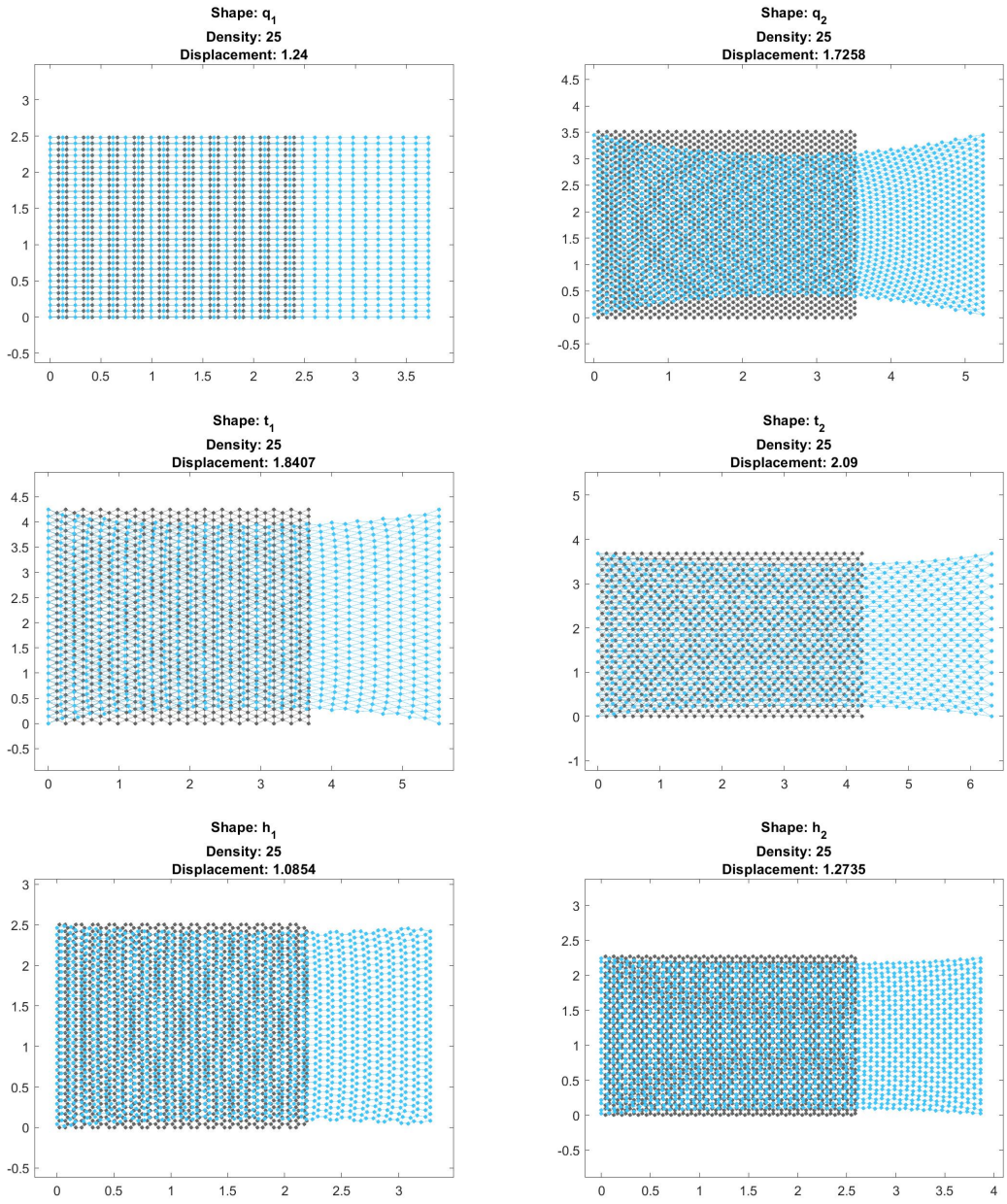

Figure 4.1: Steel networks subject to $u_x = 0.5L_0$ at density $d = 25$, with junction volume V_1.

in values of $V_1 = 2.5 \times 10^{-7}, V_1 = 1.6667 \times 10^{-7}$ and $V_1 = 3.3333 \times 10^{-7}$, respectively. For densities of $20, 30$ and 40 exemplary grid displacements for all patterns are shown in Figure 4.2.

The results of the stiffness analysis with respect to increasing density are shown in Figure 4.3. Network q_1 turns out to have the highest stiffness. This is, however, only due to vertical edges, which make up half of all edges and are not deformed. As soon as the quadratic pattern is rotated the stiffness decreases significantly. This can be seen when considering the graph corresponding to q_2, which shows the lowest stiffness values of all grids. For the triangular and hexagonal patterns, both

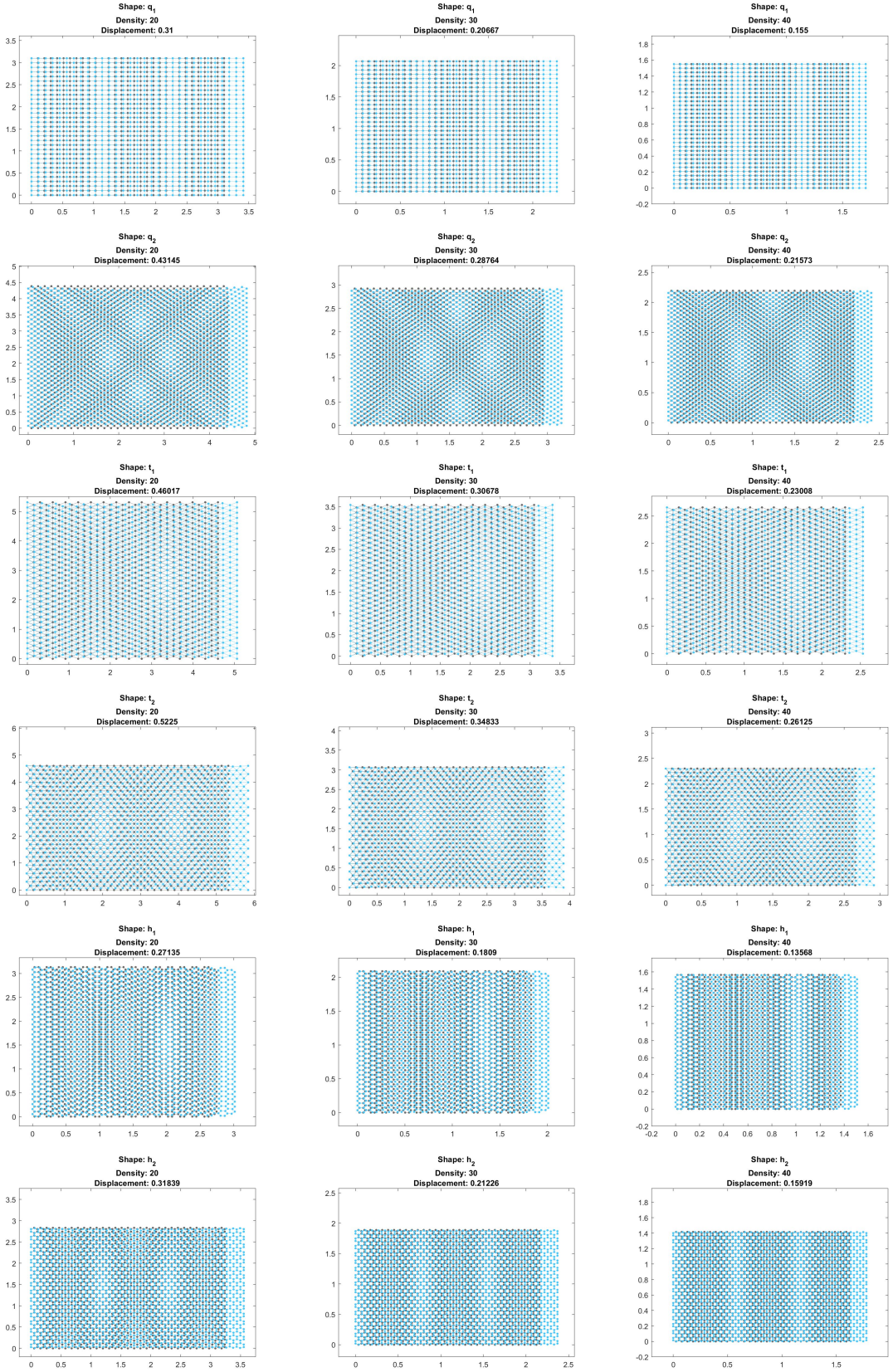

Figure 4.2: Steel networks subject to $u_x = 0.1L_0$ at densities $d = 20, d = 30$ and $d = 40$ with junction volume V_1.

orientations result in very similar stiffness values. The patterns t_1 and h_2 which include vertical edges are slightly stiffer. Surprisingly, the values of the hexagonal meshes exceed those of the t_1 and t_2 from a density of approximately $d \approx 19$ and $d \approx 17$, respectively. One explanation for this could be the chosen junction volume, since the gluing parameter for a node in the hexagonal mesh splits into only three edge pairs in contrast to six edge pairs that meet in the triangular meshes.

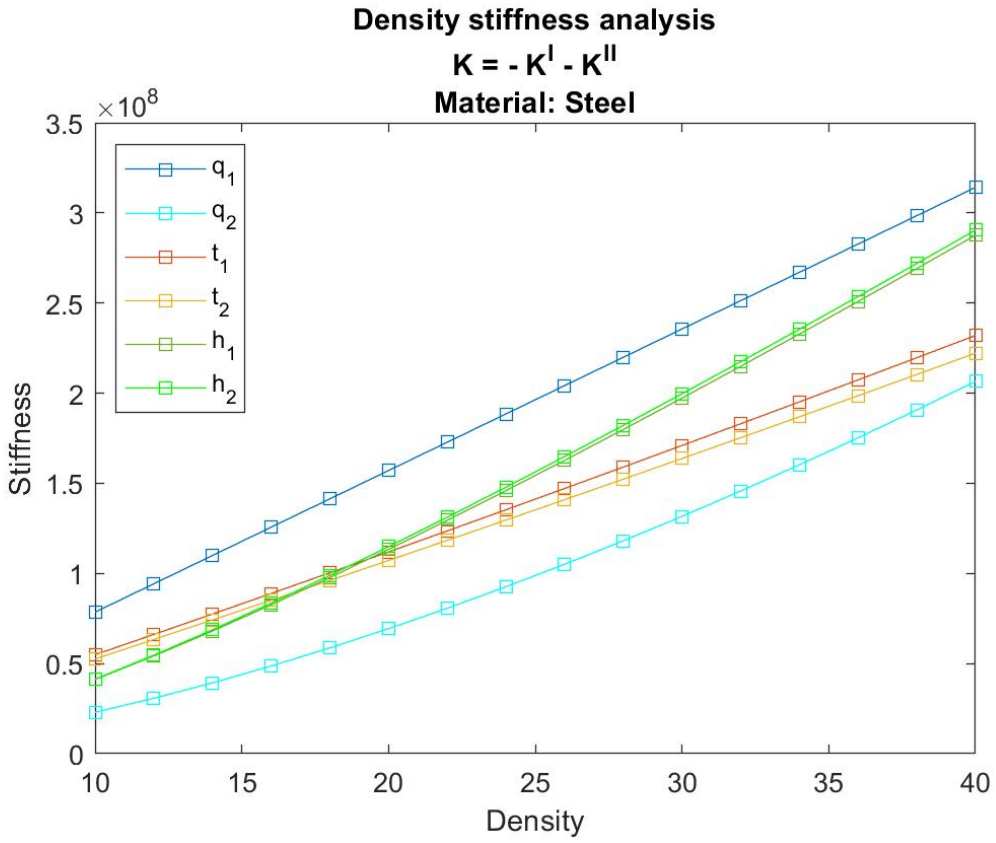

Figure 4.3: Density-stiffness analysis of all grids with 30 rows and columns, junction volume V_1 and steel parameters.

It should also be mentioned, that the edge length a gets much shorter for the hexagonal networks at higher densities than they do for the triangular grids. Related to the edge length the condition numbers of K_{BC}, denoted by $\text{cond}(K_{BC})$ are significantly higher for h_1 and h_2 than those of t_1 and t_2 at all densities. Here K_{BC} denotes the elasticity matrix adjusted to fulfill the boundary conditions again. The condition number specifies the stability of the solution for $K_{BC} \cdot u = \hat{F}$. The large values indicate that small input changes can cause huge output differences in the linear system of equations [1]. Both, the minimum and maximum edge lengths are listed in Table 4.1. Besides, the increase of the condition numbers for shorter edge lengths is illustrated in Figure 4.4.

Table 4.1: Minimum and maximum values of edge lengths a for steel networks.

Pattern	min(a)	max(a)
q_1	0.05167	0.2067
q_2	0.05171	0.2068
t_1	0.08856	0.3542
t_2	0.08856	0.3542
h_1	0.03015	0.1206
h_2	0.0313	0.1205

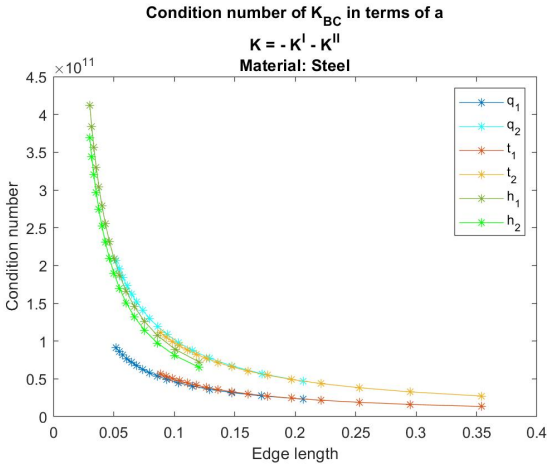

Figure 4.4: Condition numbers of K_{BC} of steel networks with junction volume V_1.

4.1.1 Variation of Junction Volume V_{ijl}

Formula (2.17) expresses the second approach to the junction volume V_2, which depends on the edge length a. Therefore, it changes with the density of the networks. The *moment of inertia* for prismatic bars of a solid circular cross section A_c, as they are considered here, reads

$$I = \frac{\pi}{4}\left(\frac{w}{2}\right)^4,$$

where w is the diameter of the bars. Steel rods with $w = 0.01$, and densities within $D = [40 : 90]$ are considered. The resulting scopes for V_2 with respect to the pattern of the grids can be found in Table 4.2. This shows that V_2 takes on much smaller values than V_1 overall.

Table 4.2: Minimum and maximum values of V_2 for steel networks with densities ranging from 40 to 90.

Pattern	min(V_2)	max(V_2)
q_1	$0.95008 \cdot 10^{-8}$	$2.1377 \cdot 10^{-8}$
q_2	$0.94930 \cdot 10^{-8}$	$2.1359 \cdot 10^{-8}$
t_1	$0.55429 \cdot 10^{-8}$	$1.2471 \cdot 10^{-8}$
t_2	$0.55429 \cdot 10^{-8}$	$1.2471 \cdot 10^{-8}$
h_1	$1.6281 \cdot 10^{-8}$	$3.6632 \cdot 10^{-8}$
h_2	$1.6289 \cdot 10^{-8}$	$3.6651 \cdot 10^{-8}$

It could not be completely clarified how the junction volume influences the stiffness. However, Figure 4.5 contains the repetition of an extreme displacement of steel networks this time using V_2. All other parameters are kept similar to those from

Figure 4.1. Comparing both results, it gets clear that V_2 evokes a larger lateral contraction than V_1. This effect is further discussed in Section 4.2.

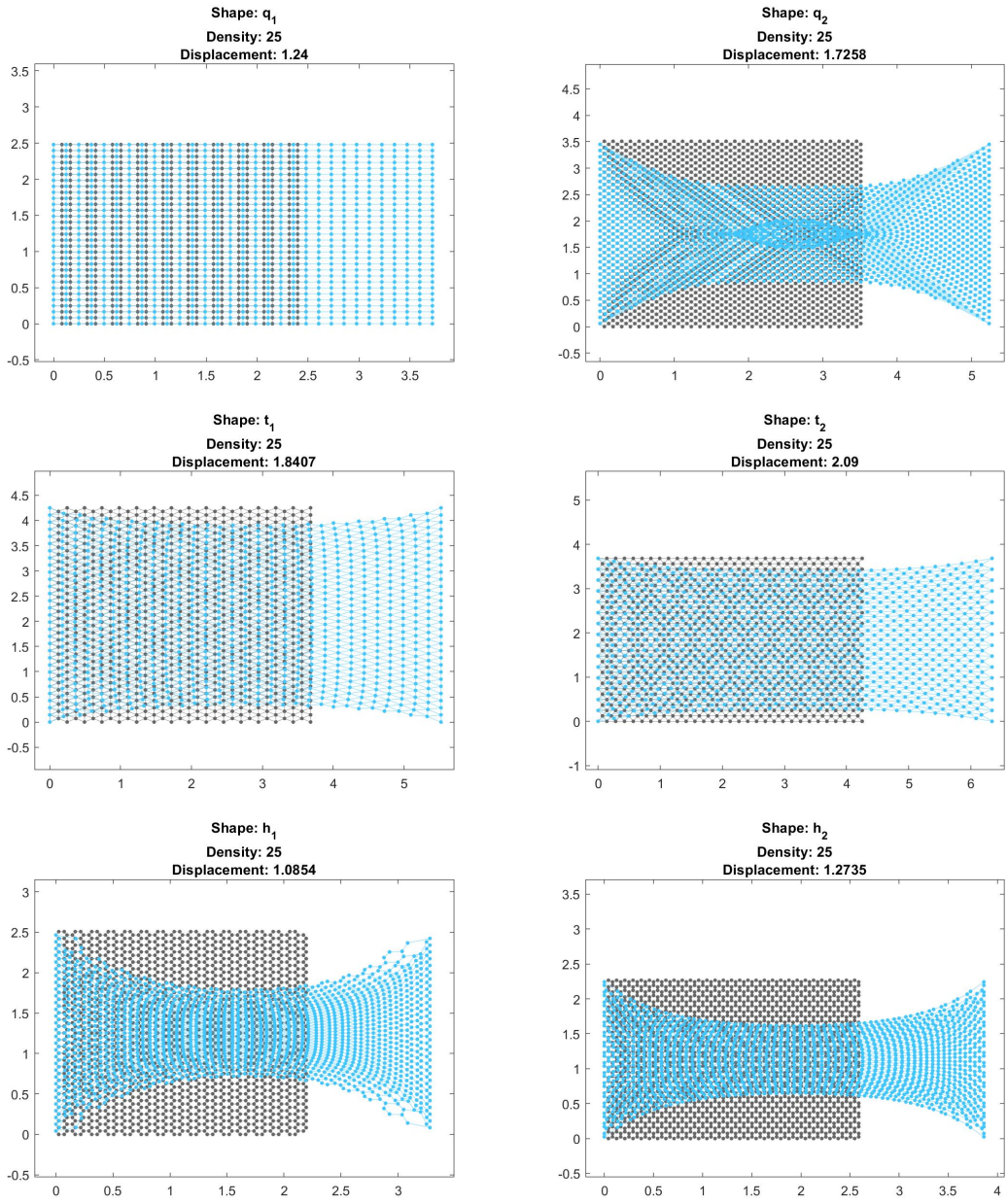

Figure 4.5: Steel networks subject to $u_x = 0.5L_0$ at density $d = 25$, with junction volume V_2.

The stiffness values obtained at the densities within D are depicted in Figure 4.6. First of all it can be noted, that the magnitudes remain similar to those presented in Figure 4.3 which correspond to the usage of V_1. Yet, clear differences are visible taking the hexagonal networks into account. In contrary to the results from Section 4.1, the stiffness values of h_1 and h_2 are clearly smaller than the results for the triangular grids. Also, q_2 takes on smaller values and the curves for all three are

much flatter. In summary the results of q_1 remain completely the same for both junction volumes. Those for t_1 and t_2 are decreasing by at most $1.67 \cdot 10^7$ and $1.70 \cdot 10^7$, which corresponds approximately 7%. This reduction is relatively small compared to the difference of more than 97% at $d = 10$ considering the hexagonal networks. Moreover, the numerical loss is larger for the hexagonal meshes and reaches a maximum of $2.3895 \cdot 10^8$ for h_2 at $d = 90$.

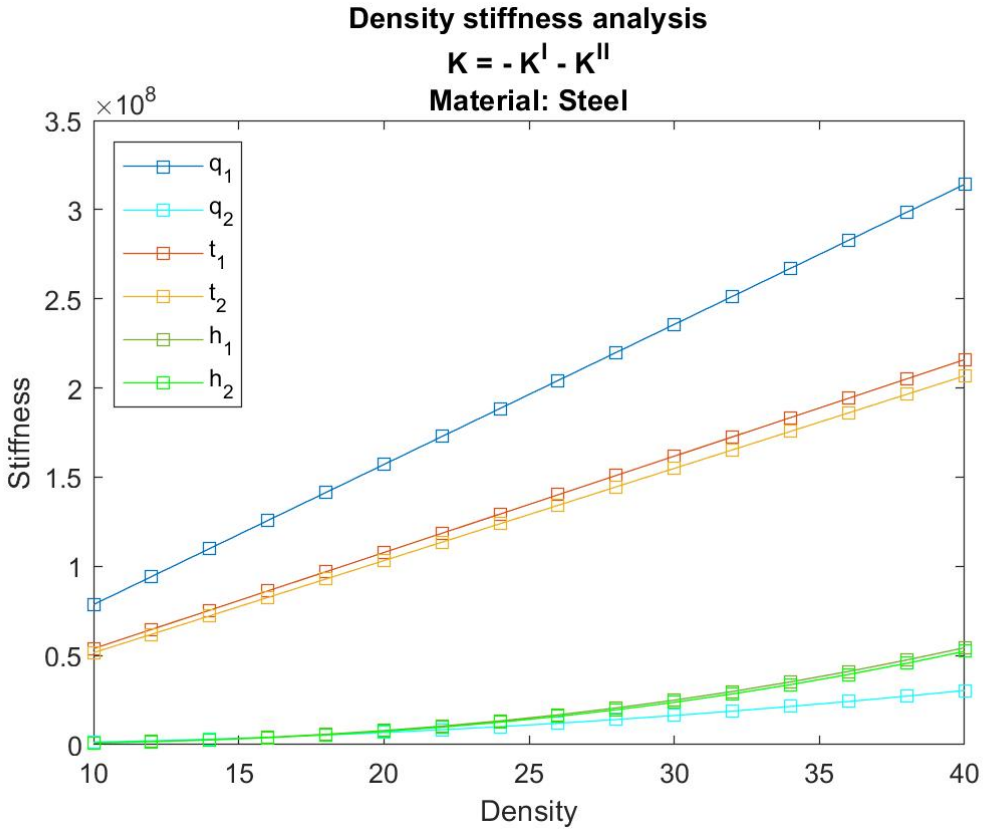

Figure 4.6: Density-stiffness analysis of all grids with 30 rows and columns, junction volume V_2 and steel parameters.

4.1.2 Variation of Bending Parameter κ

Again using the first approach to the junction volume V_1, also the bending parameter κ is iterated to investigate its effect on the stiffness development. So far, κ was equated with k and E, assuming that the angular resistance parameter equals the one describing the material's general resistance to stress.

Figure 4.7 shows the repetition of the density-stiffness analysis from Figure 4.3, with the difference, that the bending parameter is downsized to one tenth of its previous size, $\kappa = 0.1k$. The result looks very similar to Figure 4.6, obtained by exchanging V_1 with V_2. In both cases, the effect of the angular deviation gets smaller, because the factor $\frac{\kappa_{ijl} V_{ijl}}{L_{ja}}$, arising in the computation of the internal force F^{II} shrinks.

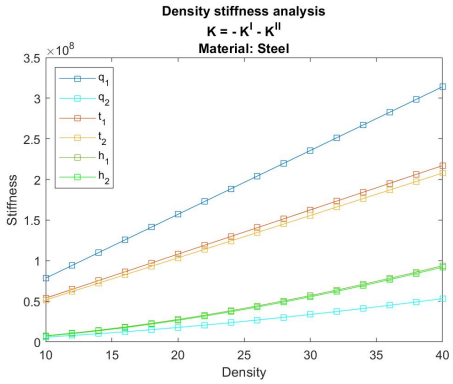

Figure 4.7: Density-stiffness analysis for steel networks with junction volume V_1 and $\kappa = 0.1k$.

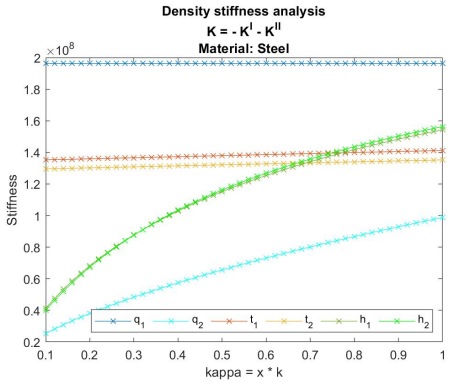

Figure 4.8: Stiffness development for steel networks at density $d = 25$, with V_1 and $\kappa \in [0.1k, k]$.

In the study presented in Figure 4.8 the bending parameter ranges between $0.1k$ and k. The stiffness of steel networks is measured at a density of 25. While the line representing q_1 is constant, those of t_1 and t_2 increase only slightly and approximately linear. In contrary, the graphs corresponding to q_1, h_1 and h_2 have roughly the form of power functions. Figure 4.8 reveals, that the stiffness values of h_1 and h_2 at $d = 25$ are overtaken by both triangular meshes for bending parameters of $\kappa = 0.66k$ or lower. From this result and the indicated results from the previous section it becomes clear that the networks h_1 and h_2 are most sensitive to angular deviation, closely followed by q_2. Whereas the results for the triangular grids remain rather equal and those of q_1 do not change at all. This observation suggests the conclusion that the angles within the hexagonal networks and q_2 change more under uniaxial stress than the angles within the meshes t_1 and t_2. Furthermore, there are no angular changes at all within the network q_1.

4.1.3 Variation of Material: Carbon and Paper

Due to the different density values required to obtain realistic fiber dimensions the comparability of studies carried out regarding different material parameters is limited. However, taking a look at the density-stiffness analysis for networks consisting of carbon and paper fibers, still some conclusions can be drawn. The necessary settings to generate the networks used for the investigations below are set according to Section 3.3.

First of all the stiffness values are much smaller than those for steel, which seems natural taking the materials into account. While those of steel range between $0.2{\cdot}10^8$ and $3.2{\cdot}10^8$ the scopes for the carbon and paper network's stiffness are just $[0.2 : 439]$ and $[62 : 2500]$, respectively. Paper occurs stiffer than carbon, even though it has a much smaller *modulus of elasticity*. That origins probably in the thickness of considered fibers, which have more than twice the diameter of carbon fibers combined with a shorter edge length. Moreover, the shapes of all stiffness-curves are approximately linear. Lastly, it stands out that the values of the hexagonal

grids are clearly below those of triangular networks. For paper and carbon fibers the values for q_2, h_1 and h_2 have a bigger distance to the stiffer networks q_1, t_1 and t_2 than is the case for steel.

4.2 Poisson's Ratio

As a second entity of interest *Poisson's Ratio*, γ, is investigated. The theoretical foundation can be found in Section 2.1.3, where the ratio is defined for a bar that gets stretched along one axis and therefore contracts in its width. Here the two dimensional networks are considered as the body of interest. The axial strain ε, is the ratio of the x-displacement u_x and the original length L_0. The lateral contraction ϵ is computed by the maximal decrease of height u_y, i.e. the maximal y-displacement, divided by the original height h_0. To assure that the maximal y-displacement equals in the positive and negative direction, it was necessary to have symmetric grids regarding the y-axis, which was resolved according to the description in Section 3.3. The resulting formula for γ is:

$$\gamma = \frac{u_y L_0}{h_0 u_x}.$$

It seems natural, that γ develops contrary to the stiffness of the networks. The stiffer the network the smaller is the maximum y-displacement and therefore *Poisson's ratio*. The grid q_1 is hardly compressed in the y-direction, which results in infinitesimal values of γ. For all other grids γ decreases with increasing density.

As indicated in Figure 4.9 the biggest *Poisson's ratio* using V_1 is obtained by network q_2, which matches the lowest stiffness value of q_2 in Figure 4.3. The triangular grids lead to almost constant and linearly decreasing values. The graphs corresponding to hexagonal grids decrease much steeper. They follow a nonlinear function and undercut the triangular ratios around density values of 17 and 19. These are the same densities for which the stiffness of h_1 and h_2 exceeds that of t_1 and t_2.

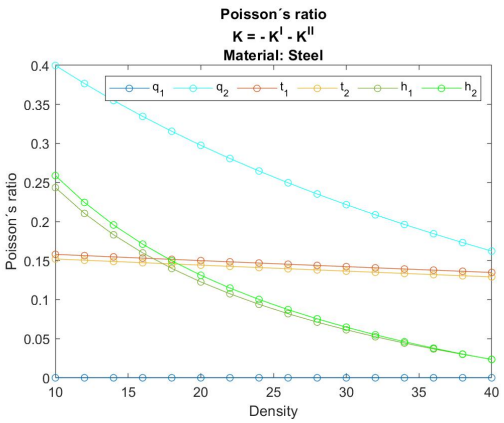

Figure 4.9: *Poisson's ratio* for steel networks, with junction volume V_1.

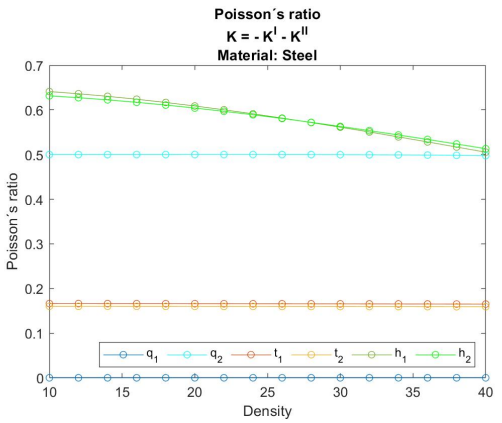

Figure 4.10: *Poisson's ratio* for steel networks, with junction volume V_2.

The overall values of the *Poisson's ratios* obtained with V_2, displayed in Figure 4.10, are higher than those using the glue parameter V_1. As it was mentioned before, this is reflected in Figure 4.5 in the more extreme reduction in height. The graphs corresponding to q_1, t_1 and t_2 are similar to those in Figure 4.9, just the triangle's ratio increases slightly by about 0.01 to 0.03. *Poisson's ratio* of q_2 with respect to V_2 decays almost linear and much slower than the one for V_1. At $d = 10$ it begins approximately 0.1 above the corresponding entry in Figure 4.9. The curve ends around 0.5 for a density of 40, which is almost 0.34 more than the value for V_1. Similar to the comparison of the stiffness values for the two different junction volumes in Figures 4.3 and 4.6, the biggest change is visible in the hexagonal grids. The corresponding graphs have an opposite curvature and the difference attains up to 0.5 at densities 28 and 32 for h_1 and h_2, respectively.

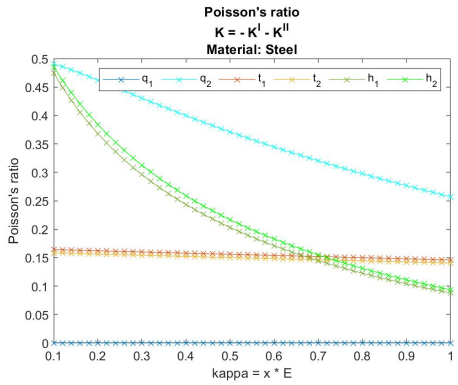

Figure 4.11: *Poisson's ratio* for steel networks at density $d = 25$, with V_1 and $\kappa \in [0.1k, k]$.

Figure 4.11 shows the repetition of the variation of κ within $[0.1k, k]$, this time with respect to *Poisson's ratio*. The graphs reflect higher values, the smaller the bending parameter gets. One exception is q_1, which shows no difference in the value of its infinitesimal small *Poisson's ratio*. In general the slopes resemble those of Figure 4.9, where instead of bending parameter the density was increased. Hence, the hexagonal grids are affected more by the variation of κ than the triangular grids. The curve of q_2 obtains the highest values overall. It descends non-linearly and quite steeply. *Poisson's ratio* of q_2, h_1 and h_2 even approach 0.5 at small κ. This indicates a plastic deformation of the object according to Section 4.2, if the resistance to angular change within those networks gets too small.

4.3 Random Manipulation of Regular Grids

Within this section the effect on the stiffness values of the networks subject to random irregularities is thematised. The experiments presented below are again carried out with the material choice of steel, the glue parameter V_1 as junction volume, and a bending parameter equal to the stiffness parameter, $\kappa = k$.

The tests are computationally heavy and therefore limited to only one choice for the density value, namely $d = 25$. As it is the mean value of the density interval presented for steel in Section 3.3. Moreover, the displacement u_x towards the right is set to 10% of the original edge length of the non-manipulated grid.

4.3.1 Node Displacement

The presented regular fiber networks could be used for a discrete model of materials, that are homogeneous, isotropic and elastic. They also work to simulate actual fabrics consisting of a grid of micro-fibers. However, their regular patterns are far from the realistic fiber arrangements, such as those produced in paper manufacturing. To simulate more realistic compositions, for example in comparison to the microscopic structure of paper, the nodes of the networks are slightly displaced. Therefore, some factor β ranging from 0 to 0.5 is chosen. Then a random displacement vector $\delta_i^\beta = (u_{x_i}^\beta, u_{y_i}^\beta)$ with values within the interval of $[-\beta a, \beta a]$ is determined for every node $i \in \mathcal{N}$. Here a describes again the unit edge length of the grid. Figure 4.12 shows an exemplary effect of this manipulation, based on network q_1.

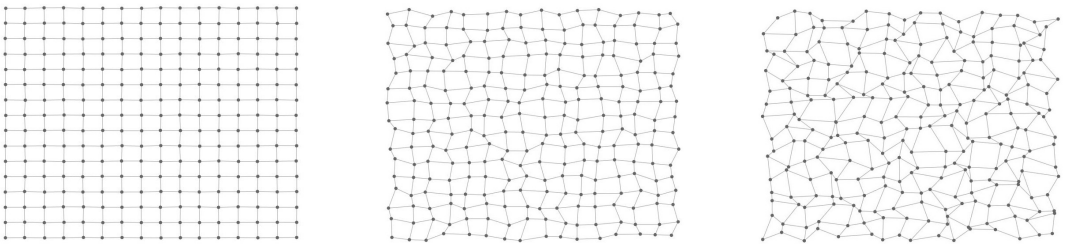

Figure 4.12: Examples of randomly displaced nodes of network q_1. The displacement factors from left to right are $\beta = 0.02, \beta = 0.25$ and $\beta = 0.5$, respectively.

In order to analyse the effect of nodal displacement for every grid pattern, the displacement range is iterated. Then the stiffness of the resulting network is compared to the original, non-manipulated mesh. In each iteration step β is increased by 0.02 and 100 networks with displaced nodes are generated. The stiffness values of those networks are scattered with blue dots in Figure 4.17. The average of them is marked red, whereas the original stiffness is illustrated by the green line. It is displayed that the deviations of the stiffness of the manipulated nets from the average value tend to become larger with increasing β. Among the samples from Figure 4.17, the largest deviation of $69 \cdot 10^6$ for q_1 is reached at $\beta = 0.42$, albeit the magnitude of the deviation does not differ significantly between the grids.

For a better comparison the average stiffness of all manipulated meshes are presented in Figure 4.13. The values at $x = 0$ display the original stiffness for steel networks at $d = 25$ with junction volume V_1. Rotation of the quadratic shape influences the stiffness by almost a factor of two (1.9874), letting q_1 appear stiffest among all shapes. Also the hexagonal shapes appear stiffer as the triangular shapes and q_2.

However, with increasing β, the original patterns get more and more deformed. Figures 4.13 and 4.17 show, that the stiffness values for all networks decline, except for q_2, which is increasing up to 3.47%. Among the original grids, q_2 shows the lowest stiffness. The growth can be explained by the fact that the edges, which are

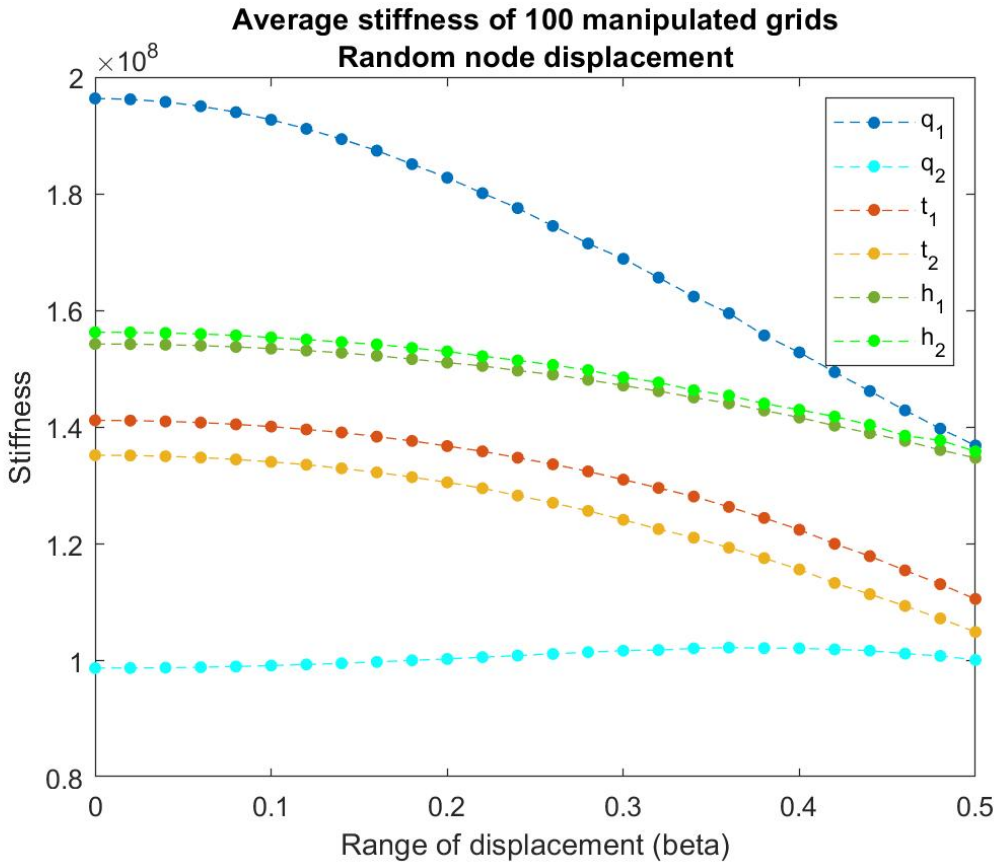

Figure 4.13: Average stiffness with increasing range of random nodal displacement in steel networks at $d = 25$ with junction volume V_1.

originally at an angle of 45° to the y-axis, approach a more vertical position and are therefore more resistant against tensile force. Yet, for displacements higher than $0.4a$ the positive effect shrinks again. The decay for all other grids follows approximately quadratic functions. Compared to the initial stiffness value of $1.9877 \cdot 10^8$ the decline to an average stiffness value of $1.369 \cdot 10^8$ at a displacement of $0.5a$ for the network q_1 marks the biggest decay among the networks with approximately 30.26%. The huge effect results from the fact that half of the original edges were vertically aligned and therefore not stretched, which they are now. It is noticeable that the hexagonal grids are effected less than q_1 and the triangular meshes. The displaced triangular meshes show a decline in stiffness of approximately 22%, whereas the loss for h_1 and h_2 is only about 13%. The average stiffness values thus fall most steeply for the hexagonal grids, followed by q_1, q_2 and lastly the triangular patterns.

To illustrate the described results of node displacement on the stiffness and therefore on the stretching behaviour of the networks subject to tensile stress, the two extreme cases of q_1 and q_2 were picked out. Figure 4.14 shows the networks, set up as described at the beginning of this section. The only difference is, that the force f_x applied on the nodes at the right hand side was fixed. For $\beta = 0.02a, \beta = 0.25a$ and $\beta = 0.5a$ the displacement was obtained with the previously computed stiffness. To

increase the visibility of the effect, a rather extreme value of $2 \cdot 10^8$ was chosen for f_x. It is shown that the displacement of q_1 is bigger, the more nodes are displaced. On the contrary, the displacement of q_2 is highest for the smallest displacement and lowest for a displacement within $[-0.25a, 0.25a]$.

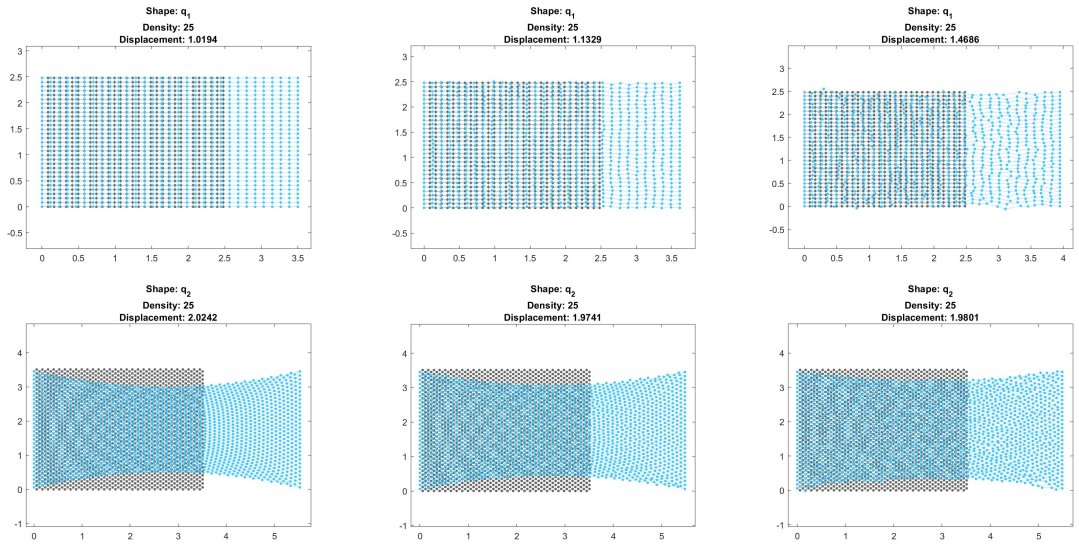

Figure 4.14: Examples of deformed networks q_1 and q_2 with randomly displaced nodes. The displacement factors from left to right are $\beta = 0.02, \beta = 0.25$ and $\beta = 0.5$, respectively.

4.3.2 Edge Removal

The second kind of change applied to the grids is the removal of edges. This measure simulates the occurrence of cracks or cuts in the networks. In order to analyse the impact of erased edges on the stiffness values up to 3% of the total number of edges are randomly chosen and erased from the set of edges $\mathcal{E}$. Thereafter the edge pairs including the chosen edges have to be discharged from $\mathcal{P}$. It is tested if the networks are still connected, i.e. if every node is still indicent to at least one edge, otherwise the manipulated network gets dismissed and the manipulation is repeated with another set of edges. In Figure 4.15 one grid of each shape with 3% of removed edges are shown as examples.

Figure 4.18 contains the results of the analysis. In each iteration the percentage of removed edges is raised by 0.1%. Again 100 samples are investigated per step, whose stiffness values are plotted with blue dots, whereas their average value and the original stiffness are marked by red dots and by the green line, respectively.

The removal of edges leads to a smaller stiffness for all patterns without exception. Moreover, the averages of the measured values decay in an approximately linear manner. According to the experiment's results, the hexagonal networks are most sensitive to the erasure of single fibers with a decline of more than 15% for both orientations. The quadratic meshes follow, since the stiffness of q_1 and q_2 falls by

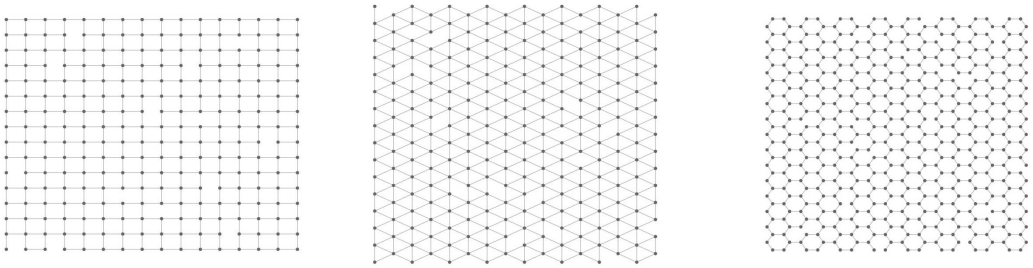

Figure 4.15: Examples of 3% randomly removed edges in q_1, t_1 and h_1 from left to right, respectively.

approximately 13.25% and 10.43%. Most resistance to cuts is displayed by the triangular grids, whose stiffness falls slightly more than 8%.

Since the highest effect is obtained for network h_2, this pattern is chosen to visualise the difference between the increasing displacement effect of $1\%, 2\%$ and 3% of removed edges in Figure 4.16. Equally to the previous procedure the force on the right hand side is set to 2×10^8. The displacement rises with the percentage from 1.5273 over 1.653 to 1.7118.

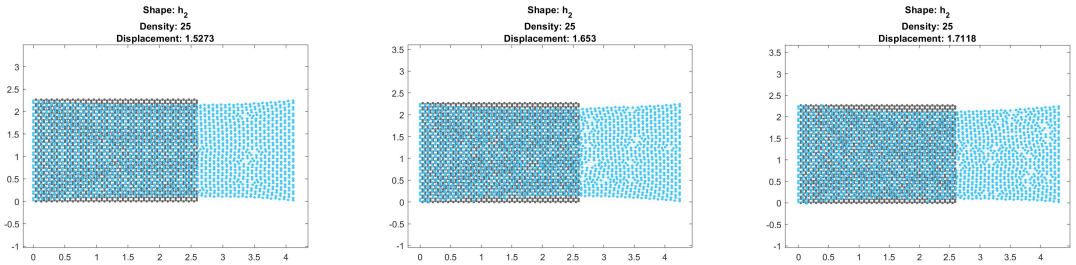

Figure 4.16: Examples of deformed network h_2 with $1\%, 2\%$ and 3% of randomly removed edges subject to a load of $f_x = 2 \cdot 10^8$.

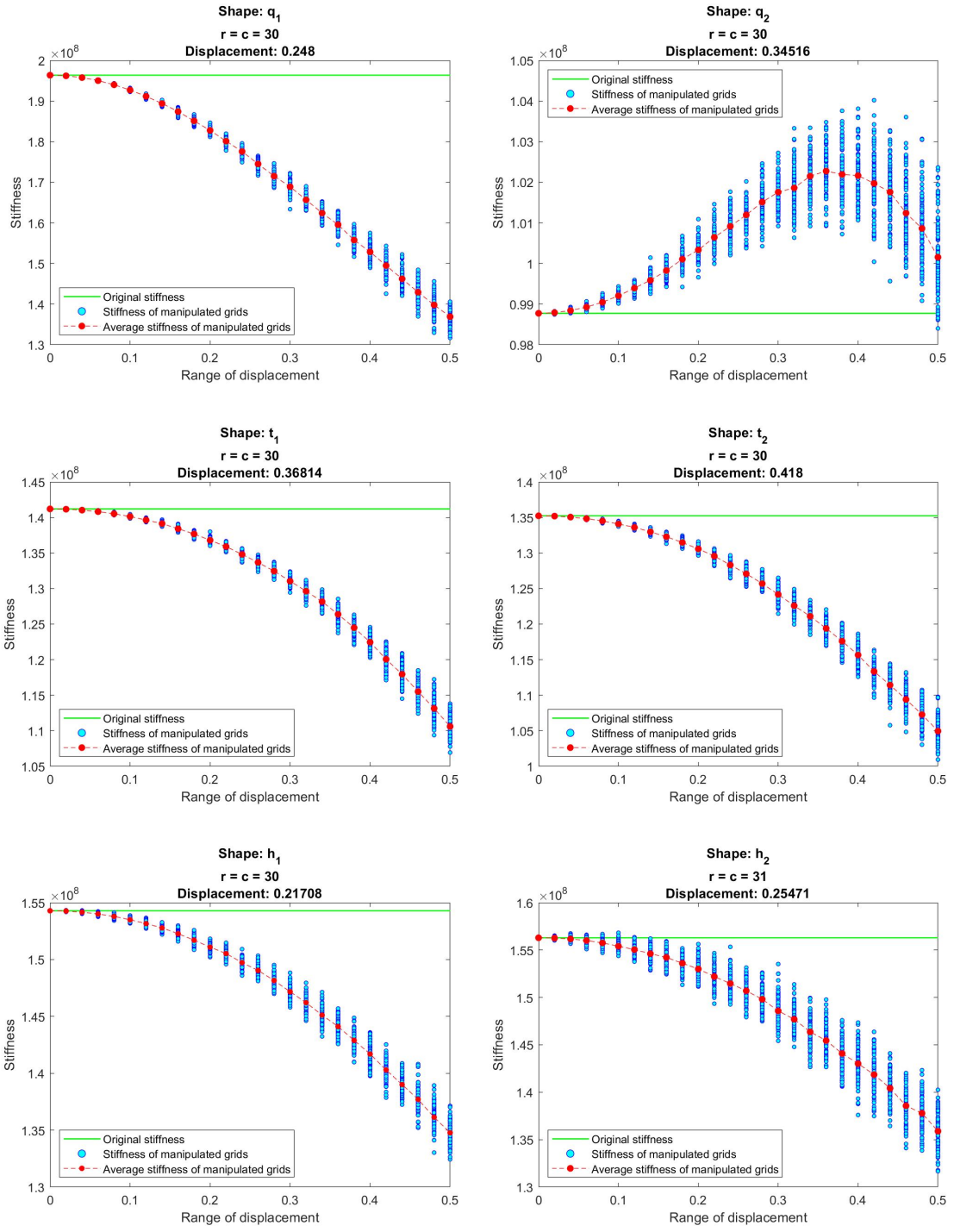

Figure 4.17: Evolving error with increasing range of random nodal displacement in steel networks of density 25 with junction volume V_1.

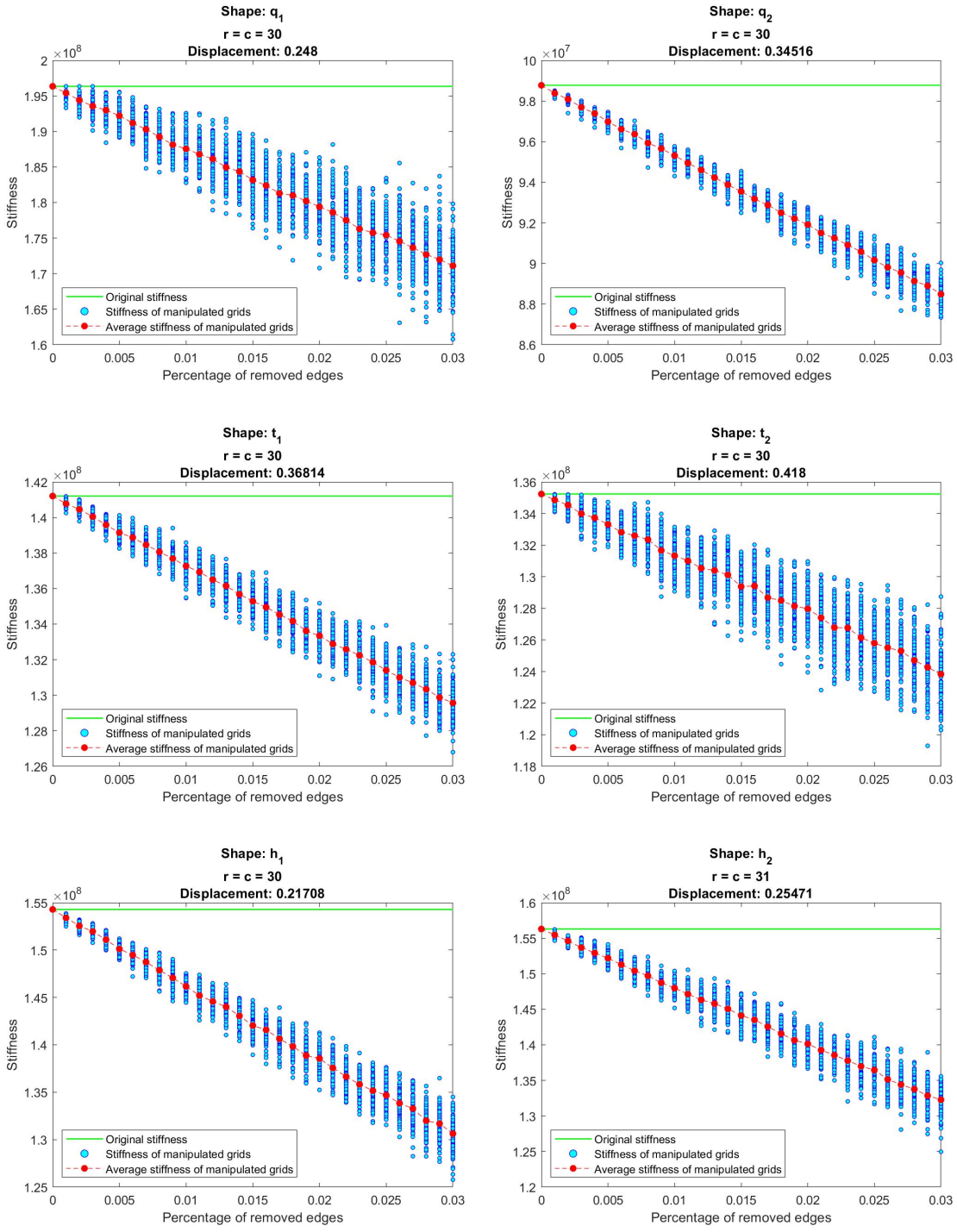

Figure 4.18: Evolving error with increasing percentage of randomly removed edges in steel networks of density 25 with junction volume V_1.

5

Conclusion & Outlook

This thesis dedicated itself to model and compare six different discrete fiber networks subject to tensile load. The grids vary in their basic regular shapes and their orientation. Several settings for the junction volume, the bending parameter and the materials were investigated. In addition, the previously regular patterns were disturbed by random changes. Even though the quadratic pattern with parallel edges to the axes ranks highest in all stiffness-investigations, it becomes clear that this shape is not the most stable, since the second quadratic setup, rotated by 45% always comes last. Taking both orientations of the patterns into account, the hexagonal shapes appear stiffest for mechanical properties of steel, a constant glue parameter used as junction volume and a bending parameter equal to *Young's modulus*. Moreover, the hexagonal meshes show most resistance towards random nodal displacement. However, changing the junction volume to one that origins in beam theory or scaling the bending parameter down adequately to the density, the triangular networks overtake the hexagonal grids in terms of their stiffness. Also, the simulation of cuts evoked least effect on the triangular meshes. Among the two triangular setups the stiffness of the grid with vertical edges outperforms the other one slightly in all density-stiffness analyses. The comparison of the network's *Poisson's ratio* revealed an opposite development of the curves with rising density compared to the stiffness. Networks with higher stiffness showed lower values of *Poisson's ratio* and vice versa.

The presented model of the fiber networks is simplified in many ways. Concluding the thesis, further steps are discussed, that might improve the accuracy of the solutions. First of all, the third internal force emerging within the fibers due to the *Poisson effect*, was completely omitted when running the experiments in Chapter 4. Therefore, one further step could be to determine the relation of the stiffness parameter η_{ijl} for the edge pairs to the stiffness parameter k_{ij} for the single edges. That would allow to adjust the settings for both parameters and to reinforce K^{III} in the assembly of the elasticity matrix K.

Secondly, as it was seen in Section 4.1.1, the choice for the junction volume significantly influences the stiffness values, at least for some patterns. Compared to using V_1 the election of V_2 even changes the order of the investigated patterns regarding their stiffness. Throughout this work it was not verified, if there is a correct or incorrect value for V_{ijl}. Also, the ratio between the size of the junction volume and the results were not determined.

Another possible extension of the model is to add the third spatial dimension. This would allow to investigate deformations of the networks in further detail. Wood fibers consists for example of a solid shell around the so-called lumen, that can collapse by stretching the fiber in its length or pressing it during the process of paper production [6], [11]. Deformations of the whole network out of the plane are imaginable especially under compressing stress which could cause creasing of the object.

Moreover, the boundary conditions provide space for possible exploration. The choice of the nodes on the right- and left hand side is natural for the network q_1 and t_1. However, for all other patterns just the nodes with the minimum and maximum x-values had been chosen. Alternatively, setting boundary conditions on all nodes at the borders of the network may be tried.

Finally, the positioning of the fibers could be more randomised to achieve a closer simulation of fibrous materials such as paper. This requires also a new way of defining the connections of edges and the edge pairs.

Bibliography

[1] L. Beilina, E. Karchevskii, and M. Karchevskii. *Numerical Linear Algebra: Theory and Applications*. Springer International Publishing, 1 edition, 2017.

[2] J.M. Gere and B.J. Goodno. *Mechanics of Materials*. Cengage Learning, 7. edition, 2009.

[3] G. Kettil, A. Målqivst, A. Mark, F. Edelvik, M. Fredlund, and K. Wester. A multiscale methodology for simulation of mechanical properties of paper. In *Proceeding of the 6th European Conference on Computational Mechanics*, Glasgow, UK, 2018.

[4] G. Kettil, A. Målqvist, A. Mark, M. Fredlund, K. Wester, and F. Edelvik. Numerical upscaling of discrete network models. *BIT Numerical Mathematics*, 60:67–92, 2020.

[5] Gustav Kettil. *Multiscale methods for simulation of paper making*. PhD thesis, Chalmers University of Technology and University of Gothenburg, 2019. Pages 57-68.

[6] R.W. Koppelaar. Properties of paper. http://mate.tue.nl/mate/pdfs/10509.pdf, 7 2009. Bachelor's Thesis, Eindhoven University of Technology. Accessed: 2021-05-06.

[7] I. Krucinska and T. Stypka. Direct measurement of the axial poisson's ratio of single carbon fibres. *Composites Science and Technology*, 41(1):1–12, 1991.

[8] P. Szymczak M. Kot, H. Nagahashi. Elastic moduli of simple mass spring models. In *The Visual Computer*, volume 31 of *10*, pages 1339 – 1350. Springer Nature, 9 2015.

[9] J.-P. Masse and D. Poquillon. Mechanical behavior of entangled materials with or without cross-linked fibers. *Scripta Materialia*, 68(1):39–43, 2013. Architectured Materials.

[10] M. Ostoja-Starzewski. Lattice models in micromechanics. *Applied Mechanics Reviews*, 55(1):35–60, 1 2002.

[11] M. Soleimani, R. J. Hill, and T. G. M. van de Ven. Elasto-capillary collapse of circular tubes as a model for cellulosic wood fibres. *Journal of Materials Science*, 50:5337–5347, 2015.

[12] W. Szewczyk. Determination of poisson's ratio in the plane of the paper. *Fibres & Textiles in Eastern Europe*, 16(4):117–120, 2008.

[13] F. F. Udoeyo. *Structural Analysis*. Temple University Press, 2020.

A

Source Code for Matrix Assembly

Local matrix K_{ij}^{I} describing the effect of edge extension on one edge (i, j):

```
%% K_I LOCAL
function [LocalKI] = Local_K_I(posXs, posYs, posXt, posYt, k, w)
%% Local length and direction vector
L_ij = sqrt((posXs-posXt)^2+(posYs-posYt)^2);
ij = [posXt-posXs posYt-posYs]';
d_ij = ij./norm(ij);
%% Assembler edge extension
coeff = k*(w*0.5)^2*pi/L_ij;
A = coeff*d_ij(1)^2;
B = coeff*d_ij(1)*d_ij(2);
C = coeff*d_ij(2)^2;
LocalKI = [-A -B A B; -B -C B C];
end
```

Assembly of K^{I} for all edges:

```
%% Edge_extension
function [K_I] = edge_extension(N, pos, s, t, k, w)
m = size(s,2);
K_I = sparse(2*N, 2*N);
for e = 1:m
    x_s = 2*s(e)-1;
    y_s = 2*s(e);
    x_t = 2*t(e)-1;
    y_t = 2*t(e);
    Loc_K_I = Local_K_I(pos(x_s), pos(y_s), pos(x_t), pos(y_t), ...
        k, w);
    K_I([x_s y_s], [x_s y_s x_t y_t]) = K_I([x_s y_s], ...
        [x_s y_s x_t y_t])+Loc_K_I;
    K_I([x_t y_t], [x_s y_s x_t y_t]) = K_I([x_t y_t], ...
        [x_s y_s x_t y_t])-Loc_K_I;
end
end
```

Local matrix K_{ijl}^{II} describing the effect of angular deviation on one edge pair (i, j, l):

```
%% K_II LOCAL
function [LocalKII]=Local_K_II(posXi, posYi, posXj, posYj, posXl,
    posYl, kappa_ijl, V_ijl)
%% Local length and direction vector
L_ji = sqrt((posXi-posXj)^2+(posYi-posYj)^2);
L_jl = sqrt((posXl-posXj)^2+(posYl-posYj)^2);
```

```
ji = [posXi-posXj; posYi-posYj];
jl = [posXl-posXj; posYl-posYj];
d_ji = ji./norm(ji);
d_jl = jl./norm(jl);
%% Edge normals
z = [0; 0; 1];
n_ji = cross([d_ji;0],z);
n_jl = cross([-d_jl;0],z);
%% Coefficients of force vectors
coeff_i = -kappa_ijl*V_ijl/L_ji;
coeff_l = -kappa_ijl*V_ijl/L_jl;
%% Coefficients of angular change
Delta_Theta = [n_ji(1)/L_ji n_ji(2)/L_ji (-n_ji(1)/L_ji-n_jl(1)/
   L_jl) (-n_ji(2)/L_ji-n_jl(2)/L_jl) n_jl(1)/L_jl n_jl(2)/L_jl];
%% Assembler angular deviation
L_i = coeff_i*Delta_Theta.*(n_ji(1:2));
L_l = coeff_l*Delta_Theta.*(n_jl(1:2));
LocalKII=[L_i;-L_i-L_l;L_l];
end
```

Assembly of K^{II} for all edge pairs:

```
%% Angular_deviation
function [K_II] = angular_deviation(N, pos, pairs, kappa, V)
p = size(pairs, 1);
K_II = sparse(2*N, 2*N);
for e = 1:p
    x_i = 2*pairs(e,1)-1;
    y_i = 2*pairs(e,1);
    x_j = 2*pairs(e,2)-1;
    y_j = 2*pairs(e,2);
    x_l = 2*pairs(e,3)-1;
    y_l = 2*pairs(e,3);
    Loc_K_II = Local_K_II(pos(x_i), pos(y_i), pos(x_j), pos(y_j),
   pos(x_l), pos(y_l), kappa, V);
    K_II([x_i y_i x_j y_j x_l y_l], [x_i y_i x_j y_j x_l y_l]) =
   K_II([x_i y_i x_j y_j x_l y_l],[x_i y_i x_j y_j x_l y_l])+
   Loc_K_II;
end
end
```

Local matrix K_{ijl}^{III} describing the effect of the *Poisson effect* on one edge pair (i, j, l):

```
%% K_III LOCAL
function [LocalKIII]=Local_K_III(posXi, posYi, posXj, posYj, posXl,
    posYl, eta_ijl, gamma_ijl, w)
%% Local length and direction vector
L_ji = sqrt((posXi-posXj)^2+(posYi-posYj)^2);
L_jl = sqrt((posXl-posXj)^2+(posYl-posYj)^2);
ji = [posXi-posXj; posYi-posYj];
jl = [posXl-posXj; posYl-posYj];
d_ji = ji./norm(ji);
d_jl = jl./norm(jl);
%% Edge normals
z_hat = [0 0 1]';
```

```
n_ji = cross([d_ji;0],z_hat);
n_jl = cross([-d_jl;0],z_hat);
%% Coefficients of force vectors
coeff_1_i = -eta_ijl*(w*0.5)^2*pi/L_ji;
coeff_2_i = gamma_ijl*w*0.5/L_jl*norm(n_ji(1:2)'*d_jl);
coeff_1_l = -eta_ijl*(w*0.5)^2*pi/L_jl;
coeff_2_l = gamma_ijl*w*0.5/L_ji*norm(n_jl(1:2)'*d_ji);
Delta_poisson_i = coeff_1_i*[d_ji(1) d_ji(2) (-d_ji(1)-coeff_2_i*
    d_jl(1)) (-d_ji(2)-coeff_2_i*d_jl(2)) coeff_2_i*d_jl(1)
    coeff_2_i*d_jl(2)];
Delta_poisson_l = coeff_1_l*[coeff_2_l*d_ji(1) coeff_2_l*d_ji(2) (-
    d_jl(1)-coeff_2_l*d_ji(1)) (-d_jl(2)-coeff_2_l*d_ji(2)) d_jl(1)
    d_jl(2)];
%% Assemblation of KIII
P_i = Delta_poisson_i.*d_ji;
P_l = Delta_poisson_l.*d_jl;
P_j = -P_i-P_l;
LocalKIII = [P_i;P_j;P_l];
end
```

Assembly of K^{III} for all edge pairs, taking the lower number of edge pairs accumulating at nodes on the boundaries into account:

```
%% Poisson_effect different eta at boarders
function [K_III]=poisson_effect_different_eta(N, pos, pairs, eta,
    eta_borders, gamma_ijl, w, x_borders)
p=size(pairs, 1);
K_III=sparse(2*N, 2*N);
for e = 1:p
    x_i = 2*pairs(e,1)-1;
    y_i = 2*pairs(e,1);
    x_j = 2*pairs(e,2)-1;
    y_j = 2*pairs(e,2);
    x_l = 2*pairs(e,3)-1;
    y_l = 2*pairs(e,3);
    %% if central node of edge pair j lies at the boarder, change
    eta_ijl
    if any(x_borders(:) == x_j)
        eta_ijl = eta_borders;
    else
        eta_ijl = eta;
    end
    %% Assembly of KIII
    Loc_K_III = Local_K_III(pos(x_i), pos(y_i), pos(x_j), pos(y_j),
    pos(x_l), pos(y_l), eta_ijl, gamma_ijl, w);
    K_III([x_i y_i x_j y_j x_l y_l], [x_i y_i x_j y_j x_l y_l]) =
    K_III([x_i y_i x_j y_j x_l y_l],[x_i y_i x_j y_j x_l y_l]) +
    Loc_K_III;
end
end
```

Assembly of K as according to (2.24):

```
function [K] = compute_K_free(N, pos, s, t, pairs, k, w, V_ijl,
    kappa_ijl, eta_ijl, gamma_ijl)
```

```
%% Computation of K
KI = edge_extension(N, pos, s, t, k, w);
KII = angular_deviation(N, pos, pairs, kappa_ijl, V_ijl);
KIII = poisson_effect(N, pos, pairs, eta_ijl, gamma_ijl, w);

K = -KI-KII-KIII;
end
```

Generation of force vector F and the total force at the right hand side f_x according to (2.26), (2.27) and (2.28). For completeness f_y is computed as well, but it will be equal to 0 since the y-position of the nodes on the right hand side was fixed.

```
%% Computation of force vector F
function [u,force_x, force_y, K_bc] = compute_F(K,displacement, N,
    nodes_lhs, nodes_rhs)
K_bc = K;
F_bc=zeros(2*N,1);
%% Fix LHS
for i=nodes_lhs
    K_bc(2*i-1,:)=0;
    K_bc(2*i,:)=0;
    K_bc(2*i-1, 2*i-1)=1;
    K_bc(2*i, 2*i)=1;
end
%% Set BC at RHS
for i=nodes_rhs
    F_bc(2*i-1) = displacement;
    K_bc(2*i-1,:) = 0;
    K_bc(2*i-1, 2*i-1)=1;
    % fix y-positions at RHS
    K_bc(2*i, :)=0;
    K_bc(2*i, 2*i)=1;
end
%% Compute u and F
u = K_bc \ F_bc;
F = K * u;
%% Compute f_x and f_y at RHS
force_x=0;
force_y=0;
for i=nodes_rhs
    force_x=force_x+F(2*i-1);
    force_y=force_y+F(2*i);
end
```

B

Source Code to Generate Grids

Implementation of q_1:

```
%% Construction of quadratic pattern first orientation
function [N, pos, s, t, pairs]=quadratic_grid(a, c, r)
%% Nodes
N = (c+1)*(r+1);
nodes = (1:N);
%node matrix
M = reshape(nodes, (c+1), (r+1))';
% position vectors
n1 = linspace(0,a*(c+1)-a,(c+1));
X = repmat(n1,(r+1),1);
X_r = reshape(X',N,1);
n2 = linspace(0,a*(r+1)-a,(r+1)); %+y_intercept;
Y = repmat(n2',1,(c+1));
Y_r = reshape(Y',N, 1);
pos = zeros(1,2*N);
pos(1:2:end) = X_r;
pos(2:2:end) = Y_r;

%% Edges
s = [];
t = [];
for i = 1:(r+1)-1
    for j = 1:(c+1)-1
        s = [s M(i, j) M(i,j)];
        t = [t M(i+1,j) M(i,j+1)];
    end
end
% Add boundaries
s = [s M((r+1),1:(c+1)-1) M(1:(r+1)-1, (c+1))'];
t = [t M((r+1), 2:(c+1)) M(2:(r+1), (c+1))'];

%% Edge pairs
pairs = [];
for j = 0:(r+1)-2
    for i = 1:(c+1)-1
        pairs = [pairs; j*(c+1)+i j*(c+1)+i+1 j*(c+1)+i+1+(c+1);
    ...
            j*(c+1)+i+1 j*(c+1)+i j*(c+1)+i+(c+1); ...
            (j+1)*(c+1)+i (j+1)*(c+1)+i+1 (j+1)*(c+1)+i+1-(c+1);
    ...
            (j+1)*(c+1)+i+1 (j+1)*(c+1)+i (j+1)*(c+1)+i-(c+1)];
    end
end
```

Implementation of q_2:

```
%% Construction of quadratic pattern second orientation
function [N, pos, s, t, pairs]=quadratic_grid_2(a, c, r)
%% Nodes
N = (2*r+1)*c+r;
nodes =(1:N);

%position vectors
x1 = zeros(2*r+1,1);
x1(1:2:end) = 0.5*sqrt(2)*a;
X = [];

y1 = [0:0.5*a*sqrt(2):r*a*sqrt(2)];
y2 = [0.5*a*sqrt(2):a*sqrt(2):r*a*sqrt(2)];
Y = [];

for i = 1:c
    X=[X; x1+(i-1)*sqrt(2)*a];
    Y=[Y; y1'];
end

X = [X; zeros(r,1)+c*a*sqrt(2)];
Y = [Y; y2'];

pos = zeros(1,2*N);
pos(1:2:end) = X;
pos(2:2:end) = Y;

%% Edges
s = [];
t = [];
for j = 1:c
    % nodes of column
    n_o_c = [1:2*r+1]+(j-1)*(2*r+1);
    nodes_of_next_column = [1:2*r+1]+j*(2*r+1);
    s = [s n_o_c(1:end-1)];
    t = [t n_o_c(2:end)];
end
for j = 1:c-1
    n_o_c = [1:2*r+1]+(j-1)*(2*r+1);
    nodes_of_next_column = [1:2*r+1]+j*(2*r+1);
    s = [s n_o_c(1:2:end-1) n_o_c(3:2:end)];
    t = [t nodes_of_next_column(2:2:end) nodes_of_next_column(2:2:
   end)];
end
% Add boundaries
s = [s nodes(end-r+1:end) nodes(end-r+1:end)];
nodes_last_column = [1:2*r+1]+(c-1)*(2*r+1);
t = [t nodes_last_column(1:2:end-1) nodes_last_column(3:2:end)];

for j=1:length(s)
    Xs=pos(2*s(j)-1);
    Ys=pos(2*s(j));
    Xt=pos(2*t(j)-1);
    Yt=pos(2*t(j));
```

```
    L_ij=sqrt((Xs-Xt)^2+(Ys-Yt)^2);
end

%% Node matrices (rearrangement of the nodes to simplify access)
% node matrix sorted to access columns
M = reshape(nodes(1:end-r), 2*r+1 , c);
% node matrix to access rows
rows=[];
for i=1:r
    % nodes of row
    n_o_r = [];
    for j = 1:size(M,2)
        n_o_r = [n_o_r M(2*i,j) M(2*i-1,j)];
    end
    rows = [rows; n_o_r];
end
last_row = [nodes(end-r+1):nodes(end)]';
rows = [rows last_row];

%% Edge pairs
pairs = [];
% vertical pairs even columns
for i=2:size(M,1)-1
    for j=1:size(M,2)
        pairs = [pairs; M(i-1, j) M(i,j) M(i+1,j)];
    end
end
% vertical pairs odd columns
for i = 2:2:size(M,1)-1
    for j=1:size(M,2)-1
    pairs = [pairs; M(i-1, j) M(i,j+1), M(i+1,j)];
    end
end
for i = 2:2:size(M,1)-2
    for j=1:size(M,2)-1
        pairs = [pairs; M(i,j+1), M(i+1,j) M(i+1,j+1)+1];
    end
end
% horizontal pairs
for i=1:size(rows,1)
    for j=2:size(rows,2)-1
        pairs = [pairs; rows(i,j-1) rows(i,j) rows(i,j+1)];
    end
end
for i = 1:size(rows,1)-1
    for j = 2:2:size(rows,2)-1
        pairs = [pairs; rows(i, j-1) rows(i+1,j) rows(i,j+1)];
    end
    for j = 3:2:size(rows,2)-1
        pairs = [pairs; rows(i+1, j-1) rows(i,j) rows(i+1,j+1)];
    end
end
% last row
for i = 3:2:size(rows,2)-2
    pairs = [pairs; rows(end,i-2) rows(end,i-1)+2 rows(end,i); ...
        rows(end,i-1)+2 rows(end,i) rows(end,i+1)+2];
```

```
end
% last row last column
pairs=[pairs; rows(end,end-2) rows(end,end-1)+2 rows(end,end)];
% last column vertical
for i=1:r
    node = nodes(end-r+i);
    column=M(:,end);
    column(2:2:end)=[];
    pairs = [pairs; column(i) node column(i+1)];
    if i ~= r
        pairs = [pairs; node column(i+1) node+1];
    end
end
```

Implementation of t_1 and t_2. The orientation can be chosen by the input variable `orientation` $\in \{1, 2\}$:

```
%% Construction of triangular pattern both orientations
function [N, pos, s, t, pairs] = triangular_grid(a,c, r,
   orientation)
%% Nodes
N = r * c + c + r + 1 - round(c * 0.5);
%% position vector
pos = zeros(1,2*N);
x_distance = sqrt(3)*0.5*a;
pos_x = (0:x_distance:c*x_distance);
pos_y = zeros(1,c+1);
pos_y(2:2:end)=0.5*a;

s = [];
t = [];

for j=0:r-1
    nodes_of_row=[1:c+1]+j*(c+1);
    %% position vector
    pos(2*nodes_of_row-1)=pos_x;
    pos(2*nodes_of_row)=pos_y+j*a;
    %% Edges
    s=[s, nodes_of_row(1:end-1)];
    t=[t, nodes_of_row(2:end)];
end
for j=0:r-2
    nodes_of_row=[1:c+1]+j*(c+1);
    nodes_of_next_row=[1:c+1]+(j+1)*(c+1);
    nodes_in_between=nodes_of_row;
    nodes_in_between(1:2:end)=nodes_of_next_row(1:2:end);

    s=[s, nodes_of_row, nodes_in_between(1:end-1)];
    t=[t, nodes_of_next_row, nodes_in_between(2:end)];
end
nodes_upper_border = [(c+1)*r+1:N];
nodes_forelast_row = [1:c+1]+(r-1)*(c+1);
nodes_in_between_last_row = nodes_forelast_row;
nodes_in_between_last_row(1:2:end)=nodes_upper_border;

```

```
38 s=[s, nodes_in_between_last_row(1:end-1), nodes_forelast_row(1:2:
       end)];
39 t=[t, nodes_in_between_last_row(2:end), nodes_upper_border];
40
41 pos_x_upper_border=pos_x(1:2:end);
42 pos(2*nodes_upper_border-1)=pos_x_upper_border(1:size(
       nodes_upper_border,2));
43 pos(2*nodes_upper_border)=a*r;
44
45 %% turning the grid
46 if orientation == 1;
47     pos_90=zeros(1,2*N);
48     pos_90(1:2:end)=pos(2:2:end);
49     pos_90(2:2:end)=pos(1:2:end);
50     pos = pos_90;
51 end
52
53 %% Edge pairs
54 pairs=[];
55 % edge pairs around the central nodes
56 for j=1:2:r-2
57     nodes_of_row=[1:c+1]+j*(c+1);
58     reference_nodes = nodes_of_row(2:2:end-1);
59     for alpha = reference_nodes
60         pairs = [pairs; ...
61             alpha-c-1 alpha alpha-1; alpha alpha-1 alpha-c-1; alpha
       -1 alpha-c-1 alpha; ...
62             alpha alpha-1 alpha+c; alpha-1 alpha+c alpha; alpha+c
       alpha alpha-1; ...
63             alpha alpha+c alpha+c+1; alpha+c alpha+c+1 alpha; alpha
       +c+1 alpha alpha+c; ...
64             alpha alpha+c+1 alpha+c+2; alpha+c+1 alpha+c+2 alpha;
       alpha+c+2 alpha alpha+c+1; ...
65             alpha alpha+1 alpha+c+2; alpha+1 alpha+c+2 alpha; alpha
       +c+2 alpha alpha+1; ...
66             alpha-c-1 alpha alpha+1; alpha alpha+1 alpha-c-1; alpha
       +1 alpha-c-1 alpha];
67     end
68     % rightmost column for odd amount of columns
69     if (-1)^c == -1
70         alpha = nodes_of_row(end);
71         pairs = [pairs; ...
72             alpha-c-1 alpha alpha-1; alpha alpha-1 alpha-c-1; alpha
       -1 alpha-c-1 alpha; ...
73             alpha alpha-1 alpha+c; alpha-1 alpha+c alpha; alpha+c
       alpha alpha-1;...
74             alpha alpha+c alpha+c+1; alpha+c alpha+c+1 alpha; alpha
       +c+1 alpha alpha+c];
75     end
76 end
77 % triangles in between
78 for j=0:2:r-2
79     nodes_of_row=[1:c+1]+j*(c+1);
80     reference_nodes = nodes_of_row(2:2:end);
81         % first orientation
82         for alpha = reference_nodes
```

```
            pairs = [pairs; ...
                alpha alpha-1 alpha+c; alpha-1 alpha+c alpha; alpha
    +c alpha alpha-1];
        end
        % second orientation
        for alpha = reference_nodes(1:end-1);
            pairs = [pairs; ...
                alpha alpha+1 alpha+c+2; alpha+1 alpha+c+2 alpha;
    alpha+c+2 alpha alpha+1];
        end
        % rightmost column for odd amount of columns
        if (-1)^c == 1
            alpha = reference_nodes(end);
            pairs = [pairs; ...
            alpha alpha+1 alpha+c+2; alpha+1 alpha+c+2 alpha; alpha
    +c+2 alpha alpha+1];
        end
end
% upmost row
% even amount of rows
if (-1)^r == 1
    nodes_of_row=[1:c+1]+(r-1)*(c+1);
    reference_nodes = nodes_of_row(2:2:end-1);
        for i = 1:size(reference_nodes,2)
            alpha = reference_nodes(i);
            pairs = [pairs; ...
                alpha-c-1 alpha alpha-1; alpha alpha-1 alpha-c-1;
    alpha-1 alpha-c-1 alpha; ...
                alpha alpha-1 nodes_upper_border(i); alpha-1
    nodes_upper_border(i) alpha; nodes_upper_border(i) alpha alpha
    -1; ...
                alpha alpha+1 nodes_upper_border(i+1); alpha+1
    nodes_upper_border(i+1) alpha; nodes_upper_border(i+1) alpha
    alpha+1; ...
                alpha-c-1 alpha alpha+1; alpha alpha+1 alpha-c-1;
    alpha+1 alpha-c-1 alpha];
        end
        % rightmost column for odd amount of columns
        if (-1)^c == -1
            alpha = nodes_of_row(end);
            pairs = [pairs; ...
                alpha-c-1 alpha alpha-1; alpha alpha-1 alpha-c-1;
    alpha-1 alpha-c-1 alpha; ...
                alpha alpha-1 nodes_upper_border(end); alpha-1
    nodes_upper_border(end) alpha; nodes_upper_border(end) alpha
    alpha-1];
        end
% odd amount of rows
else
    nodes_of_row=[1:c+1]+(r-1)*(c+1);
    reference_nodes = nodes_of_row(1:2:end-1);
        for i = 1:size(reference_nodes,2);
            alpha = reference_nodes(i);
            pairs = [pairs; ...
                alpha alpha+1 nodes_upper_border(i); alpha+1
    nodes_upper_border(i) alpha; nodes_upper_border(i) alpha alpha
```

```
    +1; ...
                ];
        end
     reference_nodes = nodes_of_row(3:2:end);
        for i = 1:size(reference_nodes,2);
            alpha = reference_nodes(i);
            pairs = [pairs; ...
                alpha alpha-1 nodes_upper_border(i+1); alpha-1
    nodes_upper_border(i+1) alpha; nodes_upper_border(i+1) alpha
    alpha-1; ...
                ];
        end
end
```

Implementation of h_1:

```
%% Construction of hexagonal pattern first orientation
function [N, pos, s, t, pairs]=hexagonal_grid_3(a,c, r)
%% Nodes
N = (c+1)*(2*r+1);
nodes = reshape([1:N], 2*r+1, c+1);
%% Position vectors
% x direction
pos_x1 = zeros(1,2*r+1);
pos_x1(1:2:end) = 0.5*a;
pos_x2 = zeros(1,2*r+1)+1.5*a;
pos_x2(2:2:end) = 2*a;
% y direction
y_distance = sqrt(3)*0.5*a;
pos_y=(0:y_distance:y_distance*(2*r));
% position vector for all nodes
pos=zeros(1,2*N);
pos(2*nodes(:,1)-1)=pos_x1;
pos(2*nodes(:,1))=pos_y;
for i=3:2:c+1
    nodes_of_column = nodes(:,i);
    pos(2*nodes_of_column-1) = pos_x1+1.5*(i-1)*a;
    pos(2*nodes_of_column) = pos_y;
end
for i=2:2:c+1
    nodes_of_column = nodes(:,i);
    pos(2*nodes_of_column-1) = pos_x2+(i-2)*1.5*a;
    pos(2*nodes_of_column) = pos_y;
end
%% Edges
s = [];
t = [];
for i = 1:c
    nodes_of_column = nodes(:,i);
    s = [s, nodes_of_column(1:end-1)'];
    t = [t, nodes_of_column(2:end)'];
end
% edges of last column
nodes_of_last_column = nodes(:,c+1);
if (-1)^c == -1
    s = [s, nodes_of_last_column(1:end-1)'];
```

```
    t = [t, nodes_of_last_column(2:end)'];
else
    s = [s, nodes_of_last_column(1:end-3)'];
    t = [t, nodes_of_last_column(2:end-2)'];
end
if (-1)^c == -1
    for i = 1:2:2*r
        nodes_of_row = nodes(i,:);
        nodes_of_next_row = nodes(i+1,:);
        s = [s, nodes_of_row(1:2:end-1) nodes_of_next_row(2:2:end-1)];
        t = [t, nodes_of_row(2:2:end) nodes_of_next_row(3:2:end)];
    end
else
    for i = 1:2:2*r
        nodes_of_row = nodes(i,:);
        nodes_of_next_row = nodes(i+1,:);
        s = [s, nodes_of_row(1:2:end-2) nodes_of_next_row(2:2:end-2)];
        t = [t, nodes_of_row(2:2:end-1) nodes_of_next_row(3:2:end-1)];
    end
    for i = 1:2:2*r
        nodes_of_row = nodes(i,:);
        nodes_of_next_row = nodes(i+1,:);
        s = [s, nodes_of_next_row(end-1)];
        t = [t, nodes_of_row(end)];
    end
end
nodes_of_last_row = nodes(2*r+1,:);
s = [s, nodes_of_last_row(1:2:end-1)];
t = [t, nodes_of_last_row(2:2:end)];
%% Edge pairs
pairs=[];
% edge pairs on diagonal edges
for i=1:2*r-1
    for j=1:c %2:c
        pairs=[pairs; nodes(i,j) nodes(i+1,j) nodes(i+2,j)];
    end
end
% last column
if mod(c,2) == 0
    for i = 1:2*r-3
        pairs = [pairs; nodes(i,c+1) nodes(i+1,c+1) nodes(i+2,c+1)];
    end
else
    if mod(r, 2)==0
        for i = 2:2*r
            pairs = [pairs; nodes(i-1,c+1) nodes(i,c+1) nodes(i+1,c+1)];
        end
    else
        for i = 2:2*r
            pairs = [pairs; nodes(i-1,c+1) nodes(i,c+1) nodes(i+1,c+1)];
```

```
        end
    end
end
% edge pairs on horizontal edges
% every odd column
for i=1:2:2*r-1
    for j=1:2:c
        pairs = [pairs; nodes(i+1,j) nodes(i,j) nodes(i,j+1); ...
            nodes(i,j) nodes(i,j+1) nodes(i+1,j+1);...
            nodes(i+1,j) nodes(i+2,j) nodes(i+2,j+1);...
            nodes(i+2,j) nodes(i+2,j+1) nodes(i+1, j+1)];
    end
end
% every even column
if (-1)^c==-1
    for i=2:2:2*r-1
        for j=2:2:c
            pairs = [pairs; nodes(i+1,j) nodes(i,j) nodes(i,j+1);
    ...
                nodes(i,j) nodes(i,j+1) nodes(i+1,j+1);...
                nodes(i+1,j) nodes(i+2,j) nodes(i+2,j+1);...
                nodes(i+2,j) nodes(i+2,j+1) nodes(i+1, j+1)];
        end
    end
    for j=2:2:c
    % bottom row
            pairs = [pairs; nodes(1,j) nodes(2,j) nodes(2,j+1);...
                nodes(2,j) nodes(2,j+1) nodes(1, j+1)];
            % top row
            pairs = [pairs; nodes(2*r+1,j) nodes(2*r,j) nodes(2*r,j
    +1); ...
                nodes(2*r,j) nodes(2*r,j+1) nodes(2*r+1,j+1)];
    end
else
    for i=2:2:2*r-1
        for j=2:2:c-1
            pairs = [pairs; nodes(i+1,j) nodes(i,j) nodes(i,j+1);
    ...
                nodes(i,j) nodes(i,j+1) nodes(i+1,j+1);...
                nodes(i+1,j) nodes(i+2,j) nodes(i+2,j+1);...
                nodes(i+2,j) nodes(i+2,j+1) nodes(i+1, j+1)];
        end
        pairs = [pairs; nodes(i+1,c) nodes(i,c) nodes(i-1,c+1); ...
            nodes(i,c) nodes(i-1, c+1) nodes(i, c+1);...
            nodes(i+1,c) nodes(i+2,c) nodes(i+1,c+1);...
            nodes(i+2,c) nodes(i+1,c+1) nodes(i, c+1);...
            ];
    end
    for j = 2:2:c-1
        % bottom row
        pairs = [pairs; nodes(1,j) nodes(2,j) nodes(2,j+1);...
            nodes(2,j) nodes(2,j+1) nodes(1, j+1)];
        % top row
        pairs = [pairs; nodes(2*r+1,j) nodes(2*r,j) nodes(2*r,j+1);
     ...
            nodes(2*r,j) nodes(2*r,j+1) nodes(2*r+1,j+1)];
```

```
    end
    pairs = [pairs; nodes(1,c) nodes(2,c) nodes(1,c+1);...
            nodes(2*r+1,c) nodes(2*r, c) nodes(2*r-1,c+1)];
end
%% erasing redundant nodes for even amount of columns
if (-1)^c == 1
    N = N-2;
    redundant_nodes = [nodes_of_last_column(1),
   nodes_of_last_column(end)];
    indices=[2*redundant_nodes-1, 2*redundant_nodes];
    pos(indices)=[];
end
end
```

Implementation of h_2:

```
%% Construction of hexagonal pattern second orientation
function [N, pos_90, s, t, pairs]=hexagonal_grid_4(a,c, r)
%% Generating grid
[N, pos, s, t, pairs] = hexagonal_grid_3(a,r,c);
%% turning the grid
pos_90=zeros(1,2*N);
pos_90(1:2:end)=pos(2:2:end);
pos_90(2:2:end)=pos(1:2:end);
```

Program to generate the grid wanted:

```
function [N, pos, s, t, pairs] = generate_grid(shape, a, c, r)
    if strcmp(shape,'q')
        [N, pos, s, t, pairs] = quadratic_grid(a,c,r);
    elseif strcmp(shape, 'q_2')
        [N, pos, s, t, pairs] = quadratic_grid_2(a,c,r);
    elseif strcmp(shape, 't_1')
        [N, pos, s, t, pairs] = triangular_grid_4(a, c, r, 0);
    elseif strcmp(shape, 't_2')
        [N, pos, s, t, pairs] = triangular_grid_4(a, r, c, 1);
    elseif strcmp(shape, 'h_1')
        [N, pos, s, t, pairs] = hexagonal_grid_3(a,c,r);
    elseif strcmp(shape, 'h_2')
        [N, pos, s, t, pairs] = hexagonal_grid_4(a,c,r);
    end
end
```

Program to compute the dimensions of the grid and their unit edge length from a given density and pattern:

```
function [a, l, h]=size_and_a_from_density(shape, d, c, r)
%% number of edges
if strcmp(shape,'q_1')
    nr_edges = 2 * c * r + c + r;
elseif strcmp(shape, 'q_2')
    nr_edges = 4 * c * r;
elseif strcmp(shape, 't_1')
    nr_edges = 3 * c * r + r - c - 1 + round((c + 1) * 0.5);
elseif strcmp(shape, 't_2')
```

```
    nr_edges = 2 * c * r + c * ceil((r + 1) / 2) + (c - 1) * floor
   ((r + 1) / 2);
elseif strcmp(shape, 'h_1')
    nr_edges = (4 * r + r + 1) * ceil(c / 2) + r * floor(c / 2);
    if mod(c, 2) == 0
       nr_edges = nr_edges + 2 * (r - 1);
    end
elseif strcmp(shape, 'h_2')
    nr_edges = (4 * c + c + 1) * ceil(r / 2) + c * floor(r / 2);
    if mod(r, 2) == 0
       nr_edges = nr_edges + 2 * (c - 1);
    end
end
%% a, length and height
if strcmp(shape,'q_1')
    a = nr_edges / (d * c * r);
    l = a * c;
    h = a * r;
elseif strcmp(shape, 'q_2')
    a = nr_edges / (2 * d * (c - 0.5) * (r - 0.5));
    l = sqrt(2) * a * (c - 0.5);
    h = sqrt(2) * a * (r - 0.5);
elseif strcmp(shape, 't_1')
    a = nr_edges * 2 / ((sqrt(3) * r * c - sqrt(3) * 0.5 * c) * d);
    h = a * r - 0.5 * a;
    l = sqrt(3) * 0.5 * a * c;
elseif strcmp(shape, 't_2')
    a = nr_edges * 2 / (sqrt(3) * (r * c - 0.5 * r) * d);
    h = sqrt(3)* 0.5 * a * r;
    l = a * c - 0.5 * a;
elseif strcmp(shape, 'h_1')
    a = nr_edges * 4 / (d * (6 * sqrt(3) * r * c - 3 * sqrt(3) * c)
   );
    h = r * sqrt(3) * a - sqrt(3) * 0.5 * a;
    l = 1.5 * a * c;
elseif strcmp(shape, 'h_2')
    a = nr_edges * 4 / (d * (6 * sqrt(3) * r * c - 3 * sqrt(3) * r)
   );
    h = 1.5 * a * r;
    l = c * sqrt(3) * a - sqrt(3) * 0.5 * a;
end
```

Program to select nodes within a rectangle given by two of its corner nodes, necessary to select the nodes for the boundary conditions:

```
function [nodes]=select_nodes(x_lower, y_lower, x_upper, y_upper,
   pos)
% number of nodes in the network
n = size(pos,2) * 0.5;
% selecting nodes within rectangle def. by "lower" and "upper"
   corner points
nodes=[];
for i=1:n
    x_pos = pos(2*i-1);
    y_pos = pos(2*i);
```

```
9       if x_pos >= x_lower && x_pos <= x_upper && y_pos >= y_lower &&
        y_pos <= y_upper
10          nodes=[nodes i];
11      end
12  end
13  end
```

C

Source Code to Compute Mechanical Entities

Program to generate the junction volume, corresponding to the edge length with respect to `choice` $\in \{1, 2\}$ for V_1 or V_2:

```
function [V]=junction_volume(shape, choice, a, w)
if choice == 1
    if strcmp(shape,'q_1') | strcmp(shape, 'q_2')
        V = 1/4*w^3;
    elseif strcmp(shape, 't_1') | strcmp(shape, 't_2')
        V = 1/6 *w^3;
    elseif strcmp(shape, 'h_1') | strcmp(shape, 'h_2')
        V = 1/3 *w^3;
    end
elseif choice == 2
    I_circ=pi*0.25*(0.5*w)^4;
    V = I_circ/a;
end
```

Program to compute the stiffness:

```
%% Stiffness
function [S] = stiffness(displacement, force, l, h)
force_per_unit_length = force/h;
strain = displacement/l;
S = force_per_unit_length/strain;
end
```

Program to compute *Poisson's ratio*:

```
%% Poisson's ratio
function PR = poissons_ratio(l,h,u_x,u)
decrease_height = max(abs(u(2:2:end)));
PR = (decrease_height/h)/(u_x/l);
end
```

Program to set material specific parameters according to Tables 3.1 and 3.2:

```
%% Setting mechanical parameters regarding the material
% if reinforcement of KIII is wanted, this list should be
    complemented by gamma and eta (and if wanted another eta value
    for the borders)
function [k, w, A_c, kappa, D] = material_parameters(material,
    factor_kappa)
```

```
if strcmp(material, 'Steel')
    % young's modulus
    k = 200e9;
    % unit diameter
    w = 0.01;
    % scope of density
    D = [10:2:40];
elseif strcmp(material, 'Carbon')
    % young's modulus
    k = 240e9;
    % unit diameter
    w = 0.000007;
    % scope of density
    D = [25:5:90];
elseif strcmp(material, 'Paper')
    % young's modulus
    k = 105e8;
    % unit diameter
    w = 0.000018;
    % scope of density
    D = [590:55:1800];
end
% cross sectional area
A_c = (0.5*w)^2*pi;
% bending param;
kappa = factor_kappa * k;
end
```

Program to compute the actual displacement u_x from the initial length L_0 and the chosen percentage of displacement:

```
function u_x = displacement(length, percentage)
u_x = (percentage / 100) * length;
end
```

Program to plot the original and displaced network:

```
%% Plot_graph
function plot_graph(shape, d, u_x, N, pos, s, t, u)
nodes=(1:N)';
m=size(s,2);
edges=(1:m)';
pos_table = reshape(pos, [2, N])';
%% Displaced positions
pos_new = pos + u;
pos_table_new = reshape(pos_new, [2,N])';
%% Node and edge table
nodetable = table(nodes, pos_table(:,1), pos_table(:,2), '
    VariableNames', {'node', 'x position', 'y position'});
edgetable = table([s' t'], edges, 'VariableNames', {'EndNodes', '
    edge'});
%% Colors
darkGrey  = [0.4 0.4 0.4];
cyan = [0.25 0.78 1.0];
%% Original graph
```

```
G = graph(edgetable, nodetable);
figure();
axis equal
    plot(G, 'NodeColor', darkGrey, 'EdgeColor', darkGrey, 'XData',
   pos_table(:,1), 'YData', pos_table(:,2))
    %xlim([-0.2 max(pos_new)+0.2]);
    xlim([min(pos_new)-0.2 max(pos_new)+0.2]);
    ylim([min(pos_new)-0.2 max(pos_new)+0.2]);
%% Displaced graph
    hold on
    plot(G, 'NodeColor', cyan, 'EdgeColor', cyan, 'XData',
   pos_table_new(:,1), 'YData', pos_table_new(:,2))
%     legend('original grid', 'displaced grid')
    title({"Shape: " + shape +  "" ; "Density: " + d + ""; "
   Displacement: " + u_x})
end
```

D

Executive Programs

Program to define the experiment's input:

```
%% Input to compare patterns
clear all
close all
clc
% material: choose between (string, capital letter) 'Steel', '
    Carbon' and 'Paper'
material = 'Steel';
% number of rows and columns (int)
rc = 30;
% percentage of original length how much right hand side should be
% displaced (num > 0)
percentage_displacement = 10;
% kappa = factor_kappa*k
factor_kappa = 1;
% Junction volume approach: choose between 1 and 2
choice_V = 1;
% Which graphic should be plottet? Choose between
% Density - Stiffness analysis: 'SD'
% Density - Poisson's ratio : 'PD'
% Condition number of K with boundary conditions - Edge length: 'CE
    '
output = 'PD';
% List of densities which should be printed or empty list
print_grids = [];
%% Shape(s)
shapes = {'q_1', 'q_2', 't_1', 't_2', 'h_1', 'h_2'};
compare_patterns(material, rc, shapes, percentage_displacement,
    choice_V, factor_kappa, output, print_grids)
```

Program to generate plots for comparing different networks at rising densities:

```
%% Compare networks of chosen shapes
% generating plots for comparison and possibly exemplary deformed
    networks
function compare_patterns(material, rc, shapes,
    percentage_displacement, choice_V, factor_kappa, output,
    print_grids)
%% Mechanical parameters
[k, w, A_c, kappa, D] = material_parameters(material, factor_kappa)
    ;
%% Generate grid and compute displacement
for shape = shapes
    % initialisation
```

```
    Stiffness = [];
    Poissons_ratio = [];
    Cond_nrs = [];
    Edge_length = [];
    for d=D
        [S, PR, Cond, a] = compute_SPrCond(k, w, A_c, kappa, d, rc,
    shape, percentage_displacement, choice_V);
        Stiffness = [Stiffness, S];
        Poissons_ratio = [Poissons_ratio, PR];
        Cond_nrs =[Cond_nrs, Cond];
        Edge_length = [Edge_length, a];
        %% Print output densities
        for j = print_grids
            if d == j
                plot_graph(shape, d, u_x, N, pos, s, t, u');
                hold off
            end
        end
    end
%% Plot
j = length(print_grids)*6;
if strcmp(output, 'SD')
    figure(j+1)
    plot(D, Stiffness, '-s')
    hold on
elseif strcmp(output, 'PD')
    figure(j+2)
    plot(D, Poissons_ratio, '-o', 'MarkerSize',5)
    hold on
elseif strcmp(output, 'CE')
    figure(j+3)
    plot(Edge_length, Cond_nrs, '-*')
    hold on
end
end
%% Title and legend of the figure
if strcmp(output, 'SD')
    xlabel('Density')
    ylabel('Stiffness')
    legend(shapes, 'Location', 'northwest')
    title({'Density stiffness analysis'; 'K = - K^I - K^{II}'; "
   Material: " + material})
elseif strcmp(output, 'PD')
    xlabel('Density')
    ylabel('Poissons ratio')
    legend(shapes, 'Location', 'northeast')
    title({'Poissons ratio'; 'K = - K^I - K^{II}'; "Material: " +
   material})
elseif strcmp(output, 'CE')
    xlabel('Edge length')
    ylabel('Condition number')
    legend(shapes, 'Location', 'northeast')
    title({"Condition number of K_{BC} in terms of a"; 'K = - K^I -
    K^{II}'; "Material: " + material})
end
```

Program to compute a network's stiffness, *Poisson's ratio* and $cond(K_{BC})$:

```
%% Compute stiffness, Poisson's ratio and condition number of a
    network
% program to ensure symetrical grids computing all entities of
    interest
% working for one shape and one density value
function [S, PR, Cond, a] = compute_SPrCond(k , w, A_c, kappa, d,
    rc, shape, percentage_displacement, choice_V)
%% Columns and rows
c = rc;
r = rc;
% ensuring symmetric grids
if strcmp(shape, 't_2') && mod(rc,2) ~= 0
    c = c+1;
    r = r+1;
elseif strcmp(shape, 'h_2') && mod(rc,2) == 0
    c = c+1;
    r = r+1;
end
% edge length and dimensions
[a, l, h] = size_and_a_from_density(shape, d, c, r);
% junction volume
V = junction_volume(shape, choice_V, a, w);
% actual displacement from percentage of displacement
u_x = displacement(l, percentage_displacement);
%% Net
[N, pos, s, t, pairs] = generate_grid(shape,a,r,c);
%% BC nodes
min_x=min(pos(1:2:end));
min_y=min(pos(2:2:end));
max_x=max(pos(1:2:end));
max_y=max(pos(2:2:end));
nodes_lhs = select_nodes(min_x, min_y, min_x, max_y, pos);
nodes_rhs = select_nodes(max_x, min_y, max_x, max_y, pos);
%% Elsasticity matrix
Edge_extension = - edge_extension(N, pos, s, t, k, w);
Angular_deviation = - angular_deviation(N, pos, pairs, kappa, V);
    %%% possibly reinforcing KIII
    %     nodes_top = select_nodes(min_x, max_y, max_x, max_y, pos)
    ;
    %     nodes_bottom = select_nodes(min_x, min_y, max_x, min_y,
    pos);
    %     nodes_boarders = [nodes_top, nodes_bottom];
    %     Poisson_effect = - poisson_effect_different_eta(N, pos,
    pairs, eta, eta_boarders, gamma, w, [2*nodes_boarders-1]);
K = Edge_extension + Angular_deviation; % + Poisson_effect
%% Displacement and force
[u ,force_x, ~, K_bc] = compute_F(K, u_x, N, nodes_lhs, nodes_rhs);
S = stiffness(u_x,force_x,l,h);
PR = poissons_ratio(l,h,u_x,u);
Cond = condest(K_bc);
end
```

Input for random manipulation:

```
%% Input for random grid manipulation
%material: choose between (string, capital letter) 'Steel', 'Carbon' and 'Paper'
material = 'Steel';
% number of rows and columns (int)
rc = 30;
% shapes: choose one or more of (string) 'q_1', 'q_2', 't_1', 't_2', 'h_1', 'h_2'
shapes = {'h_1','q_1'};
% density (int >0)
density = 25;
% percentage of original length how much right hand side should be displaced (num > 0)
percentage_displacement = 10;
% kappa = factor_kappa*k
factor_kappa = 1;
% Junction volume approach: choose between 1 and 2
choice_V = 1;
% Kind of manipulation, choose between
% Random node displacement: 'Node_displacement'
% Random removal of edges: 'Edge_removal'
manipulation = 'Node_displacement';
% manipulation range: range for beta or the percentage of removed edges
% in the thesis: [0.02:0.02:0.5] for 'Node_displacement'
% and [0:0.001:0.03] for 'Edge_removal'
manipulation_range = [0.1:0.1:0.5];
% iterations: number of samples per beta/percentage
iterations = 10;

%% Calling the program rdn_manipulation for each shape
for shape = shapes
    [S_av] = rdn_manipulation(material, rc, shape, density, percentage_displacement, choice_V, factor_kappa, manipulation, manipulation_range, iterations);
end
```

Program compare stiffness values at increasing random manipulation:

```
function [S_av] = rdn_manipulation(material, rc, shape, density, percentage_displacement, choice_V, factor_kappa, manipulation, manipulation_range, iterations)
%% Mechanical parameters
[k, w, A_c, kappa, D] = material_parameters(material, factor_kappa);
%% Stiffness of original grid
[S, ~, ~, ~] = compute_SPrCond(k, w, A_c, kappa, density, rc, shape, percentage_displacement, choice_V);
%% Rows and columns
if strcmp(shape, 't_2') && mod(rc,2) ~= 0
        c = rc+1;
        r = rc+1;
elseif strcmp(shape, 'h_2') && mod(rc,2) == 0
    c = rc+1;
    r = rc+1;
else
```

```
    c = rc;
    r = rc;
end
d = density;
[a, l, h] = size_and_a_from_density(shape, d, c, r);
V = junction_volume(shape, choice_V, a, w);
u_x = displacement(l, percentage_displacement);
%% Network
[N, pos, s, t, pairs] = generate_grid(shape,a,r,c);
%% BC nodes
min_x=min(pos(1:2:end));
min_y=min(pos(2:2:end));
max_x=max(pos(1:2:end));
max_y=max(pos(2:2:end));
nodes_lhs = select_nodes(min_x, min_y, min_x, max_y, pos);
nodes_rhs = select_nodes(max_x, min_y, max_x, max_y, pos);
%% Initialisation
S_av = [];
scatter_y=[];
%% Random manipulation
for beta = manipulation_range
    beta
    S_rdn = [];
    if strcmp(manipulation, 'Node_displacement')
        for i = 1:iterations
            pos_rdn = random_displacement(pos, -beta*a, beta*a);
            %% K
            K = -edge_extension(N,pos_rdn,s,t,k,w)-
   angular_deviation(N, pos_rdn, pairs, kappa, V);
            %% Displacement and force
            [u,force_x, ~,~]=compute_F(K, u_x, N, nodes_lhs,
   nodes_rhs);
            %% Stiffness
            Stiffness_rdn=stiffness(u_x, force_x, l, h);
            S_rdn = [S_rdn, Stiffness_rdn];
            scatter_y = [scatter_y, Stiffness_rdn];
        end
    elseif strcmp(manipulation, 'Edge_removal')
        unconnected_networks = 0;
        i = 1;
        while i <= iterations
            [s_rdn, t_rdn, pairs_rdn] = random_edge_removal(beta, s
   , t, pairs);
            %% Test if connected
            if TestConnected(N,s_rdn,t_rdn) == false
                unconnected_networks = unconnected_networks+1;
            else
                %% K
                K = -edge_extension(N,pos,s_rdn,t_rdn,k,w)-
   angular_deviation(N, pos, pairs_rdn, kappa, V);
                %% Displacement and force
                [u,force_x, ~,~]=compute_F(K, u_x, N, nodes_lhs,
   nodes_rhs);
                %% Stiffness
                Stiffness_rdn=stiffness(u_x, force_x, l, h);
                S_rdn = [S_rdn, Stiffness_rdn];
```

```
                scatter_y = [scatter_y, Stiffness_rdn];
                i=i+1
            end
        end
    end
    average_stiffness = mean(S_rdn);
    S_av = [S_av, average_stiffness];
end
scatter_x = repmat(manipulation_range, iterations, 1);
scatter_x = reshape(scatter_x, 1, iterations*size(
   manipulation_range,2));
%% Plot
x = manipulation_range;
y = zeros(1,size(manipulation_range,2))+S;
figure()
plot(x,y,'g-','Linewidth',1)
hold on
sz=10;
scatter(scatter_x,scatter_y, sz, 'filled', 'MarkerFaceColor', 'c',
   'MarkerEdgeColor', 'b')
plot(x,S_av, 'ro--', 'MarkerFaceColor', 'red', 'Markersize', 5)
% labels
if strcmp(manipulation, 'Node_displacement')
    xlabel('Range of displacement')
    ylabel('Stiffness')
    title({"Shape: "+shape; "r = c = "+r; "Displacement: "+u_x})
    legend('Original stiffness', 'Stiffness of manipulated grids',
   'Average stiffness of manipulated grids','Location' ,'SouthWest'
   )
    hold off
elseif strcmp(manipulation, 'Edge_removal')
    xlabel('Percentage of removed edges')
    ylabel('Stiffness')
    title({"Shape: "+shape; "r = c = "+r; "Displacement: "+u_x})
    legend('Original stiffness', 'Stiffness of manipulated grids',
   'Average stiffness of manipulated grids','Location' ,'SouthWest'
   )
    hold off
end
```

Program to generate a network with randomly displaced nodes at fixed β:

```
function [pos_rdn]=random_displacement(pos, scope_min, scope_max);
p = size(pos,2);
% vector of rdn numbers within chosen scope
r = (scope_max-scope_min).*rand(1,p) + scope_min;
% randomly manipulated position vector
pos_rdn = pos + r;
```

Program to generate a network with randomly removed edges at fixed percentage:

```
function [s_new, t_new, pairs_new]=random_edge_removal(percentage,
   s, t, pairs)
central_nodes = pairs(:,2);
m = size(s,2);
```

```
% integer number of edges to remove
nr_remove_edges = round(percentage*m);
% random indices of edges that will be removed
r = randi([1 m], nr_remove_edges, 1);
% keeping track of vanishing edge pairs by removing edges:
% search for any edge pair which contains the edge r(i)
% with endnodes s(r(i)) and t(r(i))
edge_pairs_to_remove =[];
for i = 1:nr_remove_edges
    remove_edge = r(i);
    for j = 1:size(central_nodes,1)
        if central_nodes(j) == s(remove_edge) && (pairs(j,1) == t(
    remove_edge) | pairs(j,3)==t(remove_edge))
            edge_pairs_to_remove = [edge_pairs_to_remove,j];
        end
        if central_nodes(j) == t(remove_edge) && (pairs(j,1) == s(
    remove_edge) | pairs(j,3)==s(remove_edge))
            edge_pairs_to_remove = [edge_pairs_to_remove,j];
        end
    end
end
%% removing edge pairs
pairs(edge_pairs_to_remove,:)=[];
pairs_new=pairs;
%% removing edges
s(r)=[];
t(r)=[];
s_new=s;
t_new=t;
```

Test whether the network with removed edges is still connected:

```
function B = TestConnected(N,s,t)
i=1;
while i<=N
    if any(s==i) || any(t==i)
        i=i+1;
        if i == N
            B = true;
        end
    else
        B = false;
        break
    end
end
end
```

DEPARTMENT OF MATHEMATICAL SCIENCES
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden
www.gu.se

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY